

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**FORECASTING FINANCIAL MARKETS USING
NEURAL NETWORKS: AN ANALYSIS OF
METHODS AND ACCURACY**

by

Jason E. Kutsurelis

September 1998

Principal Advisor:

Katsuaki Terasawa

Approved for public release; distribution is unlimited.

19981023 021

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE FORECASTING FINANCIAL MARKETS USING NEURAL NETWORKS: AN ANALYSIS OF METHODS AND ACCURACY		5. FUNDING NUMBERS		
6. AUTHOR(S) Kutsurelis, Jason E.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE		
13. ABSTRACT (maximum 200 words) This research examines and analyzes the use of neural networks as a forecasting tool. Specifically a neural network's ability to predict future trends of Stock Market Indices is tested. Accuracy is compared against a traditional forecasting method, multiple linear regression analysis. Finally, the probability of the model's forecast being correct is calculated using conditional probabilities. While only briefly discussing neural network theory, this research determines the feasibility and practicality of using neural networks as a forecasting tool for the individual investor. This study builds upon the work done by Edward Gately in his book Neural Networks for Financial Forecasting. This research validates the work of Gately and describes the development of a neural network that achieved a 93.3 percent probability of predicting a market rise, and an 88.07 percent probability of predicting a market drop in the S&P500. It was concluded that neural networks do have the capability to forecast financial markets and, if properly trained, the individual investor could benefit from the use of this forecasting tool.				
14. SUBJECT TERMS Neural Networks, Finance, Time Series Analysis, Forecasting, Artificial Intelligence			15. NUMBER OF PAGES 112	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

**FORECASTING FINANCIAL MARKETS USING NEURAL NETWORKS:
AN ANALYSIS OF METHODS AND ACCURACY**

Jason E. Kutsurelis
Lieutenant, United States Navy
B.S., United States Naval Academy, 1991

Submitted in partial fulfillment
of the requirements for the degree of

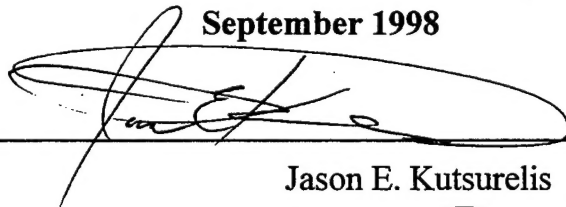
MASTER OF SCIENCE IN MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL

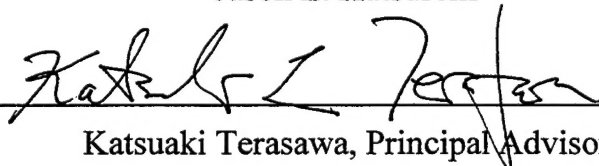
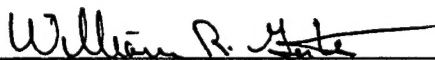
September 1998

Author:

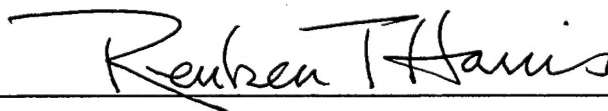


Jason E. Kutsurelis

Approved by:


Katsuaki Terasawa, Principal Advisor

William R. Gates, Associate Advisor



Reuben T. Harris, Chairman
Department of Systems Management

ABSTRACT

This research examines and analyzes the use of neural networks as a forecasting tool. Specifically a neural network's ability to predict future trends of Stock Market Indices is tested. Accuracy is compared against a traditional forecasting method, multiple linear regression analysis. Finally, the probability of the model's forecast being correct is calculated using conditional probabilities. While only briefly discussing neural network theory, this research determines the feasibility and practicality of using neural networks as a forecasting tool for the individual investor. This study builds upon the work done by Edward Gately in his book *Neural Networks for Financial Forecasting*. This research validates the work of Gately and describes the development of a neural network that achieved a 93.3 percent probability of predicting a market rise, and an 88.07 percent probability of predicting a market drop in the S&P500. It was concluded that neural networks do have the capability to forecast financial markets and, if properly trained, the individual investor could benefit from the use of this forecasting tool.

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	GOALS	1
1.	Background	2
2.	Objectives	6
3.	The Research Question	6
4.	Scope, Limitations and Assumptions.....	7
5.	Literature Review and Methodology	7
6.	Organization of Study	8
II.	LITERATURE REVIEW AND THEORETICAL FRAMEWORK	11
A.	LITERATURE REVIEW	11
B.	THEORETICAL FRAMEWORK	13
III.	METHODOLOGY	17
IV.	PRESENTATION OF DATA COLLECTED	33
A.	CLOSE NETWORK.....	33
B.	PERCENT NETWORK.....	39
C.	MULTIPLE LINEAR REGRESSION MODEL	46
D.	CONDITIONAL PROBABILITY.....	51
V.	DATA ANALYSIS AND INTERPRETATION.....	57
VI.	CONCLUSIONS AND RECOMMENDATIONS	61
A.	CONCLUSIONS.....	61
B.	RECOMMENDATIONS.....	61

APPENDIX A.	THIRD REGRESSION PARTIAL OUTPUT AND VITAL PLOTS	65
APPENDIX B.	PARTIAL LISTING OF RAW AND PREPROCESSED INPUT DATA.....	79
APPENDIX C.	SOURCE CODE FOR CLOSE NETWORK	81
LIST OF REFERENCES		95
BIBLIOGRAPHY		97
INITIAL DISTRIBUTION LIST		99

LIST OF FIGURES

Figure 1.	Biological Neuron.....	3
Figure 2.	Artificial Neuron Model	4
Figure 3.	Logistic Activation Function	5
Figure 4.	Gaussian Activation Function.....	5
Figure 5.	Artificial Neuron Using Backpropagation Learning.....	14
Figure 6.	Training Error on Close Network Training Set	34
Figure 7.	Training Error on Close Network Test Set	34
Figure 8.	Input Contribution Factors Close Network.....	36
Figure 9.	Close Network Actual vs Predicted S&P500 Close	37
Figure 10.	Close Network Error	38
Figure 11.	Training Error on Percent Network Training Set	40
Figure 12.	Training Error on Percent Network Test Set	40
Figure 13.	Input Contribution Factors Percent Network.....	41
Figure 14.	Percent Network Actual vs. Predicted Future 10 Day Percent Change and Closing Price of S&P500	42
Figure 15.	Percent Network Error	43
Figure 16.	All Network Models Actual vs. Predicted S & P500 C.....	45
Figure 17.	S&P500 Closing Value Prediction Neural Network Model Error Chart.....	46
Figure 18.	Regression Model Predicted vs. Actual S&P500 Close	48
Figure 19.	Regression Model Error Chart	49

Figure 20.	Actual vs. All Model Predictions of S&P500 Close Ten Days Into Future.....	50
Figure 21.	All Model Error Chart.....	51

LIST OF TABLES

Table 1.	Training Module Output Close Network	33
Table 2.	Close Network Output Statistics.....	35
Table 3.	Close Network Descriptive Statistics.....	38
Table 4.	Training Module Output Percent Network	39
Table 5.	Percent Network Output Statistics	40
Table 6.	Percent Network Descriptive Statistics.....	44
Table 7.	Regression Statistics	46
Table 8.	VIF for Regression.....	48
Table 9.	Regression Model Descriptive Statistics	49
Table 10.	Historical Market Data.....	51
Table 11.	Close Network Accuracy	52
Table 12.	Percent Network Accuracy	52
Table 13.	Multiple Regression Accuracy.....	53
Table 14.	Close Network Environmental Adjustment.....	53
Table 15.	Percent Network Environmental Adjustment	54
Table 16.	Multiple Regression Environmental Adjustment.....	54
Table 17.	Close Network Accuracy Probability	55
Table 18.	Percent Network Accuracy Probability	55
Table 19.	Multiple Regression Accuracy Probability	55
Table 20.	Coefficients of Multiple Determination.....	57
Table 21.	Model Error Statistics	58
Table 22.	Conditional Probabilities	59

I. INTRODUCTION

A. GOALS

This research will examine and analyze the use of neural networks as a forecasting tool. Specifically a neural network's ability to predict future trends of Stock Market Indices will be tested. Accuracy will be compared against a traditional forecasting method, multiple linear regression analysis. Finally, the probability of the model's forecast being correct will be calculated using conditional probabilities. While only briefly discussing neural network theory, this research will determine the feasibility and practicality of using neural networks as a forecasting tool for the individual investor.

The study builds upon the work done by Edward Gately in his book *Neural Networks for Financial Forecasting*. In his book, Gately (1996) describes the general methodology required to build, train, and test a neural network using commercially available software. In this research, one of Gately's S&P500 network models was validated using recent data and provided a benchmark for further improvement. Gately's model was slightly improved upon, a new model was designed, and both models were compared to a multiple regression model. Finally, and potentially most importantly for the investor, model accuracy probabilities were generated. This was done by combining historical market movement probabilities with the model accuracy probability. This conditional probability could prove to be a vital tool for investment decision making.

Until recently, neural network research, as a subset of artificial intelligence, was limited to the realm of universities, research organizations, and large investment firms.

The entrance of the neural network as an investment tool for the individual investor was one of the many things brought about by the explosive growth personal computers. Neural network software is easily available and is profusely advertised in magazines such as *Technical Analysis of Stocks and Commodities*. What isn't advertised as well is the amount of skill and effort required when building an effective model.

This research validates the work of Gately (1996) and describes the development of a neural network that achieved a 93.3 percent probability of predicting a market rise, and an 88.07 percent probability of predicting a market drop in the S&P500. It was concluded that neural networks do have the capability to forecast financial markets and, if properly trained, the individual investor could benefit from using this forecasting tool.

1. Background

Neural network theory grew out of Artificial Intelligence research, or the research in designing machines with cognitive ability. A neural network is a computer program or hardwired machine that is designed to learn in a manner similar to the human brain. Haykin (1994) describes neural networks as an adaptive machine or more specifically:

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: Knowledge is acquired by the network through a learning process and interneuron connection strengths known as synaptic weights are used to store the knowledge.

The basic building block of a brain and the neural network is the neuron. The basic human neuron adapted from Beale and Jackson (1990) is shown below in Figure 1.

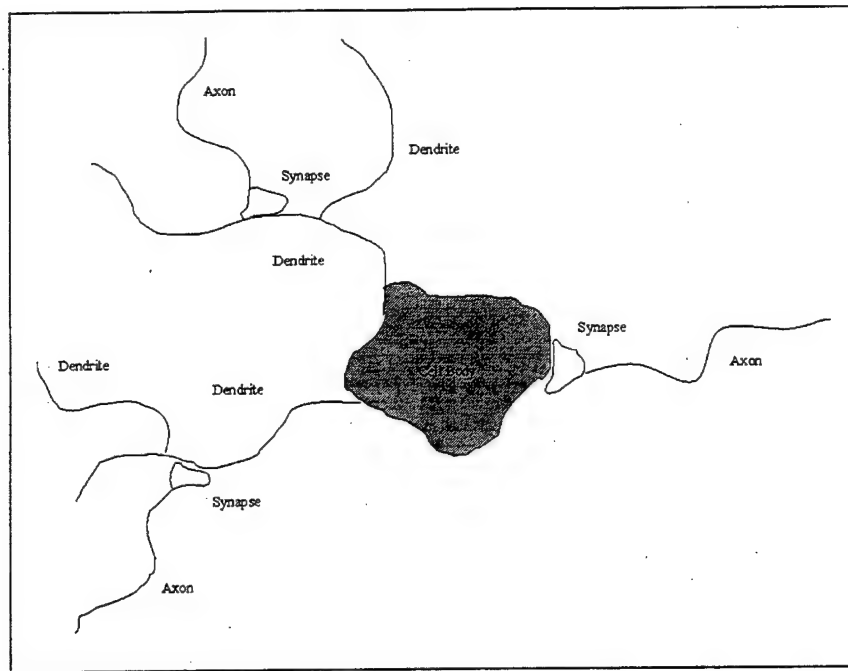


Figure 1. Biological Neuron

As described by Jackson et al. (1990), all inputs to the cell body of the neuron arrive along *dendrites*. Dendrites can also act as outputs interconnecting interneurons. Mathematically, the dendrite's function can be approximated as a summation. *Axons*, on the other hand, are found only on output cells. The axon has an electrical potential. If excited past a threshold it will transmit an electrical signal. Axons terminate at *synapses* that connect it to the dendrite of another neuron. When the electrical input to a synapse reaches a threshold, it will pass the signal through to the dendrite to which it is connected. The human brain contains approximately 10^{10} interconnected neurons creating its massively parallel computational capability.

The artificial neuron was developed in an effort to model the human neuron. The artificial neuron depicted below in Figure 2 was adapted from Kartalopoulos (1996) and Haykin (1994). Inputs enter the neuron and are multiplied by their respective synaptic

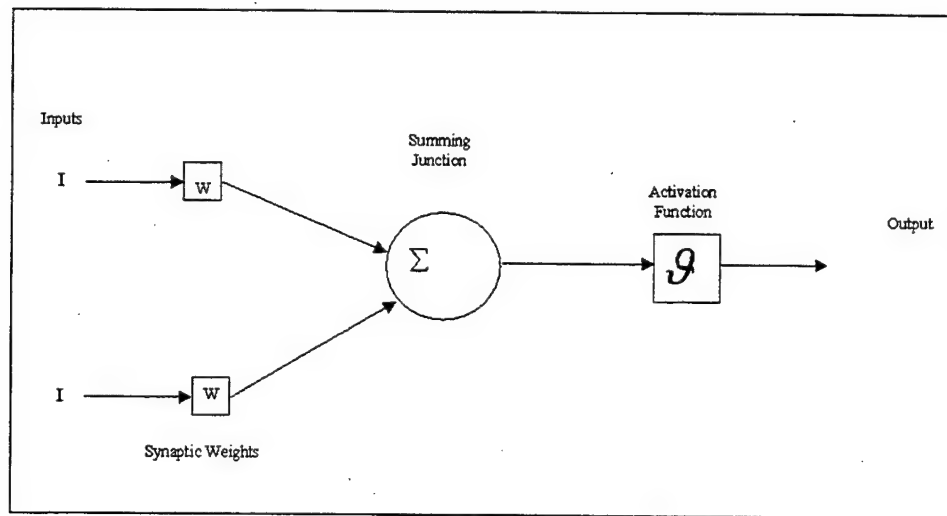


Figure 2. Artificial Neuron Model

weights. They are then summed and processed by an activation function. The activation function dampens or bound's the neuron's output, (Kartalopolous, 1996). Figures 3 and 4 represent two common activation functions which also happened to be used by the network tested during this research. The first is the logistic or sigmoid function $f(x) = 1/(1+\exp(-X))$. The second is the gaussian function $f(x) = \exp(-X^2)$.

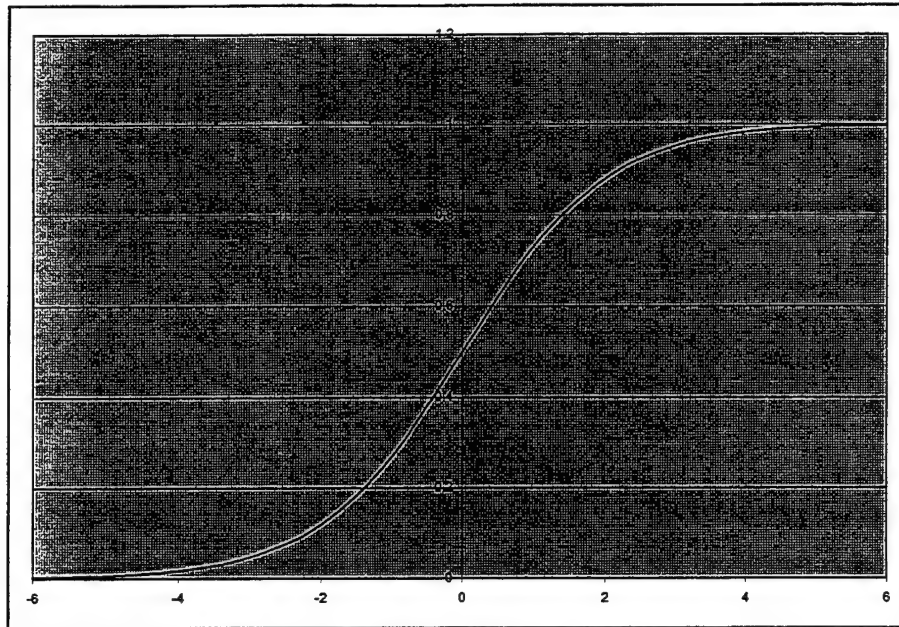


Figure 3. Logistic Activation Function

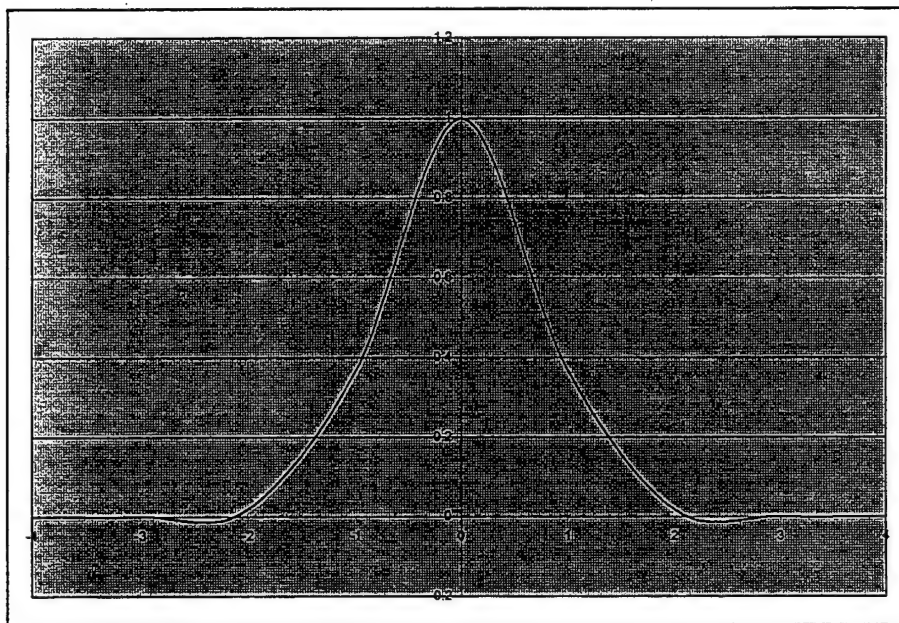


Figure 4. Gaussian Activation Function

The final output of the neuron represents the output of the activation function. Large numbers of interconnected artificial neurons have the ability to learn or store knowledge

in their synaptic weights. The learning ability of an artificial neural network will be discussed later, however this property makes it ideal for trying to identify signals within a noisy data flow. One example of such a data flow would be the closing price of the S&P500.

2. Objectives

The objective of this research was to examine the theory of Backpropagation neural networks and then develop a model that would accurately predict the future closing price of the S&P500 using a commercially available software package. Once this was accomplished, the probability of an accurate forecast would be calculated. Given the accuracy of the forecast, the benefits of the network to the investor would be determined.

3. The Research Question

The following research questions allow the research to meet the objectives proposed:

- What are the similarities between the Backpropagation neural network and the biological systems after which they were designed?
- What is the mathematical theory behind the Backpropagation neural network?
- Can neural networks accurately forecast a stock market index?
- Can multiple regression analysis accurately forecast a stock market index?
- Can neural networks be used as a practical forecasting tool by individual investors?

4. Scope, Limitations and Assumptions

The potential combinations of financial market indices or stocks, and neural network type are virtually limitless. For this reason, the research was limited to one stock market index, one neural network type and one statistical forecast tool. This allowed the research to build upon and validate previous research and place boundaries around the vast topic of time series forecasting.

The single limitation in this research was the availability of data. Gately trained his network on approximately four years of historical data. The Goldman Sachs Technology Indicator Index for Semiconductors was added to the raw data set in 1996, so the availability of data was significantly reduced. This limited the training and testing of the networks to two years of data. However, this did not seem to reduce the effectiveness of the network.

It is assumed that the reader has little or no knowledge of neural networks or statistical time series analysis.

5. Literature Review and Methodology

In his book *Neural Networks for Financial Forecasting*, Edward Gately describes the methodology needed to develop a network to accurately forecast financial markets. As an example, he develops a model that predicts the S&P500 closing value 10 days into the future. This research provides an independent validation of Gately's research and builds on it by comparing the network output to a more traditional statistical tool, multiple regression analysis. Additionally, model accuracy probabilities are examined using Bayes' Theorem.

The methodology used while conducting the research is as follows:

- Conduct a literature search of books, magazines, and the World Wide Web on the topic of neural networks.
- Identify the mathematical theory behind the Backpropagation neural networks.
- Identify a Stock Index to make forecasts upon.
- Determine what to forecast and the future point for the forecast (10 days into the future).
- Determine the inputs to the neural network using economic theory and Ward Systems Group Inc., NeuroShell 2 Professional Optimizer Package.
- Assemble the historical input data and preprocess it using Microsoft Excel.
- Using historical data, train and evaluate two different networks using Ward Systems Group Inc., NeuroShell 2 Professional.
- If needed, reassess the network inputs, retrain and evaluate the networks using Ward Systems Group Inc., NeuroShell 2 Professional.
- Attempt to forecast using multiple linear regression techniques.
- Statistically compare network forecasts vs. regression forecasts.
- Determine the historical averages of the item being forecasted occurring using Microsoft Excel and data taken from the Omega Research, Wall Street Analyst historical data CD-ROM and Dial Data Downloader.
- Determine probabilities for a successful forecast using Bayes' Theorem.

6. Organization of Study

The remaining portion of the thesis is broken up into the following chapters:

Literature Review and Theoretical Framework, Methodology, Presentation of Data Collected, Data Analysis, and Conclusions and Recommendations. Within the literature

review, pertinent literature is reviewed and the history and theory of Backpropagation neural networks is discussed. The methodology describes the steps taken to answer the research questions. Presentation of Data Collected compiles pertinent tables and charts collected during the research. Data analysis follows with a statistical and graphical review of the information presented. The thesis closes with the conclusions and recommendations.

II. LITERATURE REVIEW AND THEORETICAL FRAMEWORK

A. LITERATURE REVIEW

The potential use of neural networks as a tool for predicting financial markets has been marketed at increasing levels in recent years. Published research providing a step by step explanation of input data identification through network architecture design and finally output analysis is somewhat sparse, however. Kartalopoulos (1996), Dhar and Stein (1996), and Ward and Sherald (1995) all mention to varying degree that neural networks have the capability to forecast financial markets. Smolensky, Mozer, and Rumelhart (1996, p. 395) provide Weigand's thoughts on time series analysis and prediction. Although extremely technical, it touches on financial market prediction and provides a good overview of time series analysis. He breaks time series analysis into forecasting and modeling. Forecasting is short-term prediction while modeling tries to identify features that accurately predict long term trends. Wiegand states that these can be quite different and that the laws governing a short-term forecast may not substantially relate to the long-term model or the actual characteristics of the system.

More specifically, Weigand claims that the "...complexity of a model useful for forecasting may not be related to the actual complexity of the system." Potentially models can accurately predict markets where they are substantially less complex than the market itself. Lowe (1994) focuses on portfolio optimization and short term equity forecasting. Some believe that efficient market theory causes predictions based upon historical price patterns to be valueless. However, Lowe (1994) postulates that "A system which is apparently random could have significant deterministic components embedded in its data." He states that a neural network's "...ability to create nonlinear

approximations to the underlying generators of data...may be exploited.” Lowe (1994) concludes that it would be possible to develop an “...automated trading system based entirely upon quantitative pattern processing techniques capable of consistently outperforming professional traders.”

Lederman and Klein (1995, p. 65) provide Jurik’s thoughts on trading system development. Although Jurik does not provide specific examples of trading systems, he provides a wealth of advice on data preprocessing techniques. He states, “Strive for simple models having only a few choice input variables.” He supports this by explaining that as the number of model inputs increase, the degrees of freedom of the governing equation also increases. While equations with high degrees of freedom have the capability to model the training data effectively, they fail miserably when given test data. This is because models with fewer degrees of freedom do not try to trace the data’s random scattering but only follow the general trend. Jurik also states that “When trying to remove unimportant variables, sensitivity analysis of nonstationary or nonlinear models has dubious practical value.” This is extremely important because this is one of the standard techniques used in regression analysis. If applied to a neural network model it could seriously fail. Jurik states there are only two ways to correctly remove unimportant variables. The first is to use a genetic algorithm to develop multiple combinations of input variables while only letting the most accurate survive. The second is a manual method of systematically removing one variable at a time and recording network accuracy. This technique is repeated until the accuracy of the model starts decreasing.

While all these authors hint at the capability of neural networks in forecasting financial markets, the researcher found only one text that meticulously tracks the development of the neural network from data gathering and preprocessing to training and application of the net. This text is Edward Gately's *Neural Networks for Financial Forecasting*. This research uses Gately's (1996) techniques for developing a model to predict the closing price of the S&P500. His model is slightly improved upon and compared to more standard regression methods for forecasting. Additionally and potentially most importantly for the investor, model accuracy probabilities were generated. This was done by combining historical market movement probabilities with the model accuracy probability. This conditional probability could prove to be a vital tool for investment decision making.

B. THEORETICAL FRAMEWORK

As described previously, a neural network is a computer program or hardwired machine that is designed to learn in a manner similar to the human brain. Additionally, Figure 2 showed how an artificial neuron processes input data into an output signal. However, the most important feature of a neural network has not been explained: how a neural network learns. This research focuses on the Backpropagation algorithm learning method. The following derivation is taken from the explanation provided by Dhar and Stein (1996). All mathematical formulae refer to Figure 5 below. This figure depicts a single artificial neuron, which learns using the Backpropagation learning algorithm.

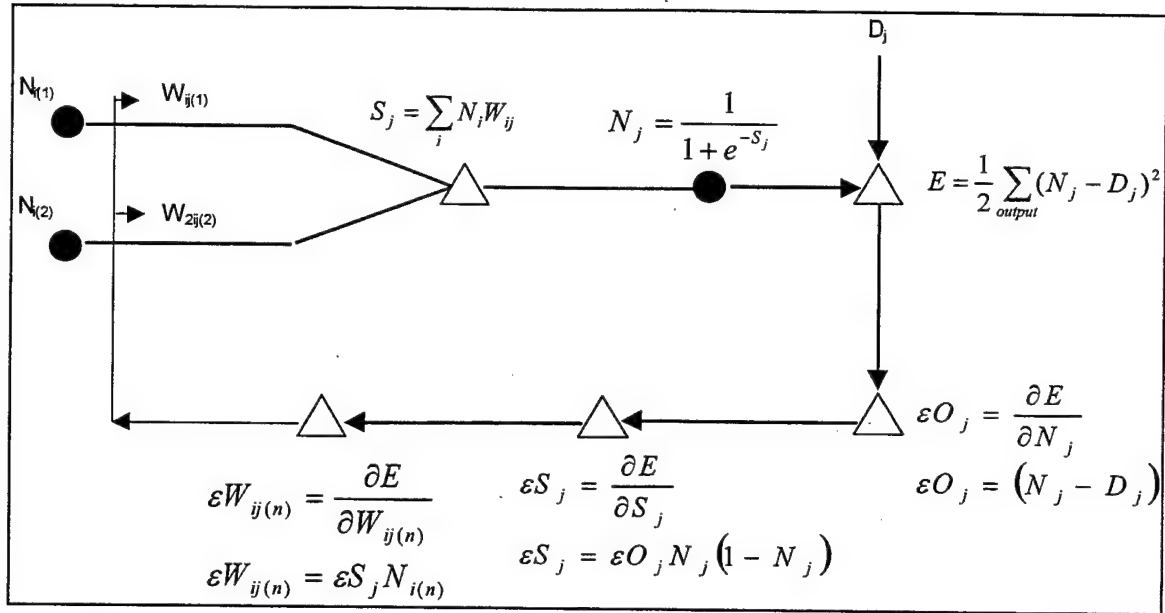


Figure 5. Artificial Neuron Using Backpropagation Learning

The Backpropagation algorithm seeks to minimize the error term between the output of the neural net and the actual desired output value. The error term is calculated by comparing the net output to the desired output and is then fed back through the network causing the synaptic weights to be changed in an effort to minimize error. The process is repeated until the error reaches a minimum value. The network uses Equation 1 to update the weight W_{ij} from a given node N_i to the current node N_j ; where t refers to the number of times the network has been updated and λ refers to the learning parameter. The learning parameter, or learning rate, controls the rate the weight is changed as learning takes place. The sensitivity of node N_j to a change in weight W_{ij} is represented by ϵW_{ij} and will be defined more fully below.

$$W_{ij,(t+1)} = W_{ij,t} + (\lambda)(\epsilon W_{ij})(N_i) \quad (1)$$

The total input to a node is described in Equation 2 as:

$$S_j = \sum_i N_i W_{ij} \quad (2)$$

where S_j is the sum of all inputs to a node, N_i is the output of the previous node, and W_{ij} is the weight connection between the i th node of the previous layer.

This output is then transformed using the logistic activation function described in chapter I and represented by Equation 3. The total output of node j is represented by N_j .

$$N_j = \frac{1}{1 + e^{-S_j}} \quad (3)$$

The overall error for a single pass of the neural network is represented by Equation 4, where D_j is the desired output of the output node j .

$$E = \frac{1}{2} \sum_{Output} (N_j - D_j)^2 \quad (4)$$

Now that the error term for the entire network has been calculated, this information is fed back through the network to reduce error. The simplified partial differential equations required to change the connection weights are provided below; the original equation before simplification can be found in Figure 5.

First, the error term for each output node O_j must be calculated. Essentially we are trying to identify how much the error term changes with respect to a change in each output node. This calculation is simplified in Equation 5 below:

$$\epsilon O_j = (N_j - D_j) \quad (5)$$

Second, the amount the error term changes as the input is varied to a given output node must be calculated. This is done by determining how much Equation 4 changes when the total input to the node (Equation 2) is changed. This calculation is simplified in Equation 6 below:

$$\epsilon S_j = \epsilon O_j N_j (1 - N_j) \quad (6)$$

Next, the weight adjustment needed for W_{ij} is calculated from a layer below the current layer (N_i) to the current node N_j . The calculation is simplified in Equation 7 below:

$$\epsilon W_{ij} = \epsilon S_j N_i \quad (7)$$

Finally this operation is continued on nodes in lower layers by allowing nodes in the lower hidden layers to play the role of the output node. All errors from all inputs to the hidden layer must be summed. Additionally, the error in the hidden node is calculated by examining how the error of nodes above the hidden node change with respect to changes in the hidden node. The simplified equation for calculating the weight update is provided in Equation 8 below, where the subscript j represents nodes in the layer above the hidden layer.

$$\epsilon H_i = \sum_j \epsilon S_j W_{ij} \quad (8)$$

Dhar et al (1996) states, "In this manner the error of the network is propagated backward recursively through the entire network and all of the weights are adjusted so as to minimize the overall network error."

III. METHODOLOGY

This section discusses the techniques used to develop the neural network and multiple regression financial forecasting models. The combination of these models, with historical and conditional probabilities, will allow the investor to make decisions based on probabilities of success. The presentation first discusses the development of the models, followed by the historical probability calculations and concludes with the conditional probability calculation.

The information to be forecast was first identified. Since this research extended and verified Gately's (1996) work with neural networks, a similar model output was chosen for at least one model for comparison purposes. Two separate model outputs were chosen, the first was the S&P 500 closing value ten days into the future—the output chosen by Gately, and the second was the future S&P500 ten day percent change. As described by Gately, a much more accurate forecast is possible when predicting the percent change versus the actual closing value of a stock or index. For example, if the S&P500 index was valued at 1400 dollars and a model that predicted the closing value had a ten-percent error the potential output inaccuracy would be 140 dollars. If the same index had made a ten-percent price change and the model that predicted the percent change of price had a ten-percent error, the potential output inaccuracy would be one-percent. For a ten-percent price change, from 1272 dollars to 1400 dollars, this inaccuracy would translate into approximately a twelve dollar error.

The first neural network model, named Close Network, was chosen to be a verification of Gately's (1996) work and predicted the S&P500 index ten days into the future. The inputs to the model were developed by first downloading the following raw data from Dial Data using Data Downloader software from Omega Research:

S&P 500 Index	*SPX
Dow Jones Transportation Index	*DWT X
Dow Jones Industrial Index	*DWI X
Dow Jones Utilities Index	*DWU
Commodities Research Bureau	*CRB
AMEX Oil Index	*XOI
Gold and Silver Mining Index	*XAU

The data set encompassed the trading days from March 1, 1991 to August 18, 1998.

The data was charted using stock charting software called Wall Street Analyst Deluxe from Omega research. This established date integrity within each index. The indices were then exported and saved as a text file from Wall Street Analyst. Using Microsoft Excel, each index text file was combined into one spreadsheet. The resulting file was formatted such that each index and its date was in a separate column with each row representing a trading day. An example of a partial data set can be found in Appendix B. Date integrity is extremely important to the neural network. To ensure integrity was maintained, each index's date column was checked for integrity against the S&P500 date using a simple "if then" rule in Excel. If the dates were equal, it placed a

zero in a row. If the dates were not equal, Excel placed a 1 in a row. This function was copied down the entire data set and then summed at the bottom. A sum of zero represented no date integrity discrepancies. A sum of anything greater than zero indicated that a date discrepancy existed. The sum was zero for all indexes with the exception of *CRB. Within *CRB it was found that one trading day was missing. Instead of deleting the index, the trading day was added. The average closing value of the day proceeding and following the missing day was chosen to represent the missing day.

The last column in the spreadsheet was titled "Future (10 day) S&P500 C." This represents the future closing price of the S&P500 Index and was the actual value the network used to compare against its prediction during training. This "future" information was created by simply copying the S&P 500 closing price into the last column of the spreadsheet deleting the first 10 days of data and moving the remaining data up 10 rows. Once this was complete, all date columns were deleted with the exception of the one associated with the S&P 500 data. The file was saved as an Excel 4 worksheet so it could be imported into Neuroshell 2 Professional. The following data was imported into the neural network software package as a pattern file:

Date

S&P500 H

S&P500 L

S&P500 C

Dow Transportation Index C

Dow Utilities Index C

Amex Oil Index C

CRB CDow Industrial Index V

Gold and Silver Mining C

Future (10 days) S&P500 C

The pattern file was then altered using the software's Market Indicator module to include all the inputs described by Gately (1996). The following technical indicators were added:

MvAvg (30) of Dow Industrial Index V

Lag (10) of CRB C

Lag (10) of Amex Oil Index C

Lag (10) of Dow Transportation Index C

These technical indicators represent a thirty-day moving average of the Dow Industrial Index Volume and a ten-day lag of various other raw data.

Within the Define Inputs module of the Neuroshell2, each column of the data file was identified either as Unused, Input, or Actual Output. The date and raw volume information was excluded by defining it as Unused. The Future (10 days) S&P500 Close was defined as the Actual Output and all others were defined as inputs. The minimum and maximum values for each data column were then automatically calculated for use by the network. Next a test set or "out of sample" data set was extracted from the data set.

Every tenth data set was chosen to represent the test data. There were 1700 training rows and 188 test rows. In the Designing the Network module, the software recommended using a Ward Network. The Ward Network is a backpropagation network with one input layer, three hidden layers and one output layer. Each layer is called a slab. Each slab is made up of one to sixteen individual neurons depending on the location within the network. Each slab also has a different activation function as described below:

Slab 1 (input): Linear [-1,1], 13 Neurons

Slab 2 (hidden): Gaussian, 16 Neurons

Slab 3 (hidden): tanh, 16 Neurons

Slab 4 (hidden): Gaussian comp., 16 Neurons

Slab 5 (output): logistic, 1 Neuron

The learning rate was set to .05 and the momentum was set to .5. Within the Training Criteria module, pattern selection was set to random, calibration interval was set to every 200 patterns and stopping criteria was set to stop training at 40,000 events since min average error within the test set. Within the training module, the network was set to automatically save the weights for the best test set. The network was then trained within the Training module. The trained network was then applied to the 188 sample test patterns.

The network output is a file that contained three columns: Actual (1), Network (1) and Act (1)-Net (1) representing the actual S&P500 closing value, the network prediction and the network error. This file was opened and actual vs. predicted charts

were created within Excel. Additionally, the descriptive statistics for the network error were calculated. The Close Network was able to make predictions with accuracy comparable to Gately's (1996) model.

The second network model, called Percent Network, was designed to predict the future S&P500 ten day percent change taking advantage of the possible reduction in prediction error. The network was provided the same raw data as the Close Network. However, two new columns were created in the raw data file. The first column was defined as the ten-day % change in the S&P500 Closing price. Starting on the 10th day, this was calculated using the following formula $(\text{cell } 1 - \text{cell } 11) / \text{cell } 1 * 100$. The second column was defined as the future ten-day % change. Essentially, the ten-day % change column was copied into the new column and the values were shifted back ten days. The future ten-day percent change was the actual desired output of the net and was used during training to calculate error.

The file was imported into Neuroshell2 and preprocessed. During preprocessing, the following technical indicators were added to the data file:

MvAvg (30) of Dow Industrial Index V

Lag (10) of CRB C

Lag (10) of Amex Oil Index C

Lag (10) of Dow Transportation Index C

LinRegChange (10,5) of S&P500 C

LinRegChange (10,10) of S&P500 C

The technical indicator LinRegChange (x,n), as described in Neuroshell2 Professional, represents "Predicted change between current value and the value time periods into the future. Prediction is based upon the linear regression line calculated from the last n time periods." The test set extraction, network design and training settings were identical to the Close Network. Essentially, the Percent Network would look at the same raw data, with added indicators, as the Close Network but would predict a percent change versus a closing price. The same out of sample data was used. When the Percent Network (1) was applied to the out of sample data, it performed relatively poorly when compared to the Close Network.

To increase the accuracy of the Percent Network, the net inputs were changed. The LinRegChange (10,5) of S&P500 C was removed and LinRegPredict (10,10) of S&P 500 10 day % Change was added. The technical indicator LinRegPredict (x,n), as described in Neuroshell2 Professional, represents the "Predicted value x time periods into the future. Prediction is based upon the linear regression line calculated from the last n time periods." After training and application to the out of sample data, Percent Network (2) showed only slight improvement in prediction accuracy.

The network inputs were again adjusted to try and increase accuracy. The technical indicator LinRegChange (10,10) of S&P500 C was removed as an input. After training and application to the out of sample data, the Percent Network (3) accuracy decreased.

At this point, the decision was made to augment the moving average of the volume with the raw volume data in the network. The Percent Network (5) performed only slightly better after including raw volume.

A close review of the raw data used in Gately's model and the Close Network showed little if any input from the technology sector. A technology component was added to the net to improve accuracy. This net was called Percent Network (5) and was based on the most accurate network up to that point, Percent Network (2). It included an added indicator, the closing value of GSM the Goldman Sachs Technology Indicator Index for Semiconductors. Unfortunately, GSM has only been in existence since 1996, so the data set was significantly reduced. The full raw data set begins on July 18, 1996 and ends August 12, 1998. Due to the reduced amount of available data, every 5th data set was chosen to represent the test or out of sample data. There were 419 training rows and 104 test rows. Percent Network (5) was trained and applied to the out of sample data. The network showed significant improvement, although the r^2 value was still not as high as the Close Network.

In Percent Network (6), the ten-day lag of GSM C was added as an input to improve prediction capability. The network was trained and applied to the out of sample data set. Although its r^2 value was slightly lower than Percent Network (5), it was chosen as the final Percent Network because it had a lower percent error over 30% and there were predictions within 5%.

The addition of the new GSM C data to the Percent Network had significantly improved its accuracy, and could do the same for the Close Network. Also the networks were using data from significantly different time periods, making comparison between the two questionable. Therefore, the GSM C and Lag of GSM C was added to the Close Network causing the data window to match the Percent Network. All data prior to July 18, 1996 was discarded. In the Extraction Module, every 5th data set was chosen to represent the test or out of sample data. Both models were now training on identical raw data. After training, the network was applied to the out of sample data. The results showed a slight decrease in r^2 value but fully 100 percent of the predictions were within five percent of the actual closing value. This network was chosen to be the final Close Network model.

The actual and predicted values from both the Close and Percent networks were placed into one spreadsheet. The predicted closing value of the S&P500 index was calculated using the predicted % change provided by the Percent Network. This allowed comparison of a single type of data between both networks. The two predicted closing values were compared graphically and statistically. This completed the neural network portion of the research.

A more traditional statistical forecasting tool is regression analysis. This method uses the sum of the least squared errors to fit a curve to a data set. The decision was made to try and predict the S&P500 closing value using the same data used in the Close Network. The dependant variable was designated as the Future (10 days) S&P500 C

column. The following columns were listed as independent variables: S&P500 H, S&P500 L, S&P500 C, Dow Transportation Index C, Dow Utilities Index C, Amex Oil Index C, CRB C, Gold and Silver Mining C, GSM C, MvAvg(30) of Dow Industrial Index V, Lag(10) of CRB C, Lag(10) of Amex Oil Index C, Lag(10) of Dow Transportation Index C, and Lag(10) of GSM C. Using the data analysis tool within Excel, a multiple linear regression analysis was performed on the data set.

When conducting multiple linear regression analysis, the following assumptions must hold for the model to be correct: 1. Normality 2. Homoscedasticity 3. Independence of Errors and 4. Linearity. Normality assumes the value of the Y (the dependant variable) must be normally distributed for each value of X (the independent variable). According to Levine, Berenson, and Stephan (1997), regression analysis is fairly robust against departures from the normality assumption. One method of verifying the normality assumption is to construct and examine a Normal Probability Plot for the dependant variable. The Normal Probability Plot was created in Excel using the regression analysis tool. The points on the plot seemed to deviate from a straight line in a random manner. This indicates normality. If the line had risen more steeply at first and then increased at a decreasing rate it would have indicated a left skewed data set. The opposite is true for a right skewed data set.

Homoscedasticity assumes variation or error around the regression line should be similar for low and high values of the independent variable. This can be verified by examining the residual plots for each independent variable. For each variable, there did

not seem to be major differences in the variability of the residual for different values of the independent variable. Therefore, the Homoscedasticity assumption was valid.

Autocorrelation, or the likelihood that a certain type of error precedes or follows another type of error, violates the independence of errors assumption. If errors are correlated, there will be a pattern of positive errors following positive errors and negative errors following negative errors. The simplest way to rule out Autocorrelation is to plot the residuals over time. The residual vs. time plot showed no pattern and indicated the absence of Autocorrelation. Therefore, the Independence of Errors assumption seemed valid.

The regression line fit plots indicated a probable linear relationship of varying degrees between the dependant variable and each of the independent variables. This was verified because there was no pattern in the residual plots for each independent variable. Therefore, the linearity assumption seemed valid.

All four assumptions of regression analysis were verified so a linear regression model seemed appropriate. When using Multiple Regression, the objective is to utilize only those variables that have a significant relationship with the dependant variable. The first step in determining a significant relationship between the dependant and independent variable was to conduct an F test. The F test was used to determine if there was a significant relationship between the dependant variable and the chosen independent variables. The null hypothesis was that there was no linear relationship between the dependent variable and indepentent variables; the alternative hypothesis was that at least one regression coefficient was not equal to zero. The null hypothesis is rejected at a

certain level of significance if the estimated value F is greater than the critical value of F . Excel's ANOVA calculation provided the value for F ; it was 1416. For a 95% confidence interval, the value of the critical F was obtained from Levin et al's (1997) Critical Values of F table. The critical value for $F(14,469)$ is approximately 1.67. Therefore, F was greater than the critical value of F and the conclusion can be made that at least one of the independent variables was related to the dependant variable.

The next step was to test individual portions of the multiple regression model. If individual variables have no significant effect on the model, they should be removed and a new regression calculated. The t test was used to determine if an individual independent variable had a significant effect on the dependant variable, taking into account the other independent variables. The null hypothesis was that there was no relationship between the independent and dependant variable; the alternative hypothesis was that there was a relationship. The decision rule was to reject the null hypothesis if the estimated t was less than negative t critical or greater than t critical. For a 95% confidence interval, the critical value for t was obtained from Levin et al's (1997) Critical t Table. The critical values for t (.025,14) were -2.1448 and 2.1448. Eight of the 14 independent variables failed the t test and did not have a significant relationship to the dependant variable. The regression was recalculated using only those independent variables that passed the t test.

At this point, all assumptions were reevaluated and found to still be valid. The F test was then repeated on the second regression model. The critical value of $F(6,477)$ is approximately 2.1. The model passed the F test. Therefore, at least one of the

explanatory variables was related to the dependant variable. The t test was then repeated on the second regression model. The critical value of $t(.025,6)$ is 2.4469. In reviewing the t values, it was determined that there is a significant relationship between the dependant variable and all independent variables with the exception of Gold and Silver Mining C. This input was dropped and a third regression was calculated.

Again all regression assumptions were reevaluated and found to be valid. The F test was then repeated on the third regression model. The critical value of F (5,477) is approximately 2.21. The model passed the F test. Therefore, at least one of the explanatory variables is related to the dependant variable. The t test was then repeated on the second regression model. The critical value of $t(.025,6)$ is 2.5706. In reviewing the t values, there was a significant relationship between the dependent variable and all independent variables.

The final test of the validity of the multiple regression model is to verify there is no multicollinearity between the independent variables. According to Levine et al (1997), when two independent variables are highly collinear they can cause the regression coefficients to fluctuate drastically if one or both are included in the model. It is difficult to separate the effect of two collinear independent variables on the dependant variable. The measure of collinearity is the Variance Inflationary Factor (VIF). For each of the independent variables, the VIF was calculated using excel. If sets of variables are uncorrelated, the VIF will equal 1. For highly intercorrelated variables, the VIF can exceed 10. According to Levine et al (1997), a VIF greater than 10 indicates there is too much correlation between the independent variables. In the third regression model, all

VIF values were less than 10. Therefore, the third regression model was fully optimized and verified and was chosen as the final model to compare to the two neural networks.

With three working forecasting models, the next step was to calculate the probability that each model's prediction will be accurate. The calculation was based upon Bayes' Theorem, or conditional probability. Bayes' Theorem states that a probability depends upon the environment in which it is based. A conditional probability is stated as given X what is the probability of Y or $P(Y|X)$. First the researcher had to find something that could easily be identified in the environment. It would be tedious to try and calculate probabilities for individual S&P500 closing values or individual percent changes. However, one could calculate the number of times the market rose or fell; this is potentially enough information for the investor. The market rise or fall corresponds to percent change, which is the output of the Percent Network. Percent changes can also be calculated from the output of the Close Network and the Regression Model.

Therefore, the question used to calculate the probability that each model's prediction is accurate is: given a certain amount of historical daily market rises, when a model predicts a market rise, what is the probability of it actually occurring. Mathematically this would be stated as $P(\text{ForecastedRise}|\text{HistoricalRise})$. Rather than using the mathematical form of Bayes' Theorem, the conditional probability was calculated in a spreadsheet using a method described by Dr. Katsuaki Terasawa of the Naval Postgraduate School. First, the historical data was calculated within Excel from the raw data during the period from March 3, 1991 to August 18, 1998. An if-then

statement was used to identify a percentage rise. The statement generates a one if the percent change is greater than zero or a zero if the percent change is less than zero. Summing this column provides the total days with a percent increase. This was subtracted from the total number of days in the data set to identify market falls. The number of times the market rose and fell was calculated.

Next, the accuracy of each model was calculated. The output of each model for the out of sample data was used. The if-then technique was used to identify when both the model forecast and the actual data agreed or disagreed as to a market rise or fall. This data could be entered into Bayes' theorem to obtain the conditional probability. However the accuracy of the model was adjusted using the historical environmental data as described by Professor Terasawa. Calculating the probability of accurately predicting a rise or fall in the S&P500 Index became a simple matter of dividing accurate rise or fall predictions by total rise or fall predictions.

IV. PRESENTATION OF DATA COLLECTED

In this section the research data will be presented and important items will be identified. The presentation begins with vital data for each of the forecasting models, follows with the historical probabilities, and concludes by presenting conditional probabilities.

A. CLOSE NETWORK

The following vital data given in Table 1 and Figures 6 and 7 is from the completed training module:

Table 1. Training Module Output Close Network

	Training Set	Test Set
Learning events	559800	N/A
Learning epochs	1446	N/A
Minimum average error	0.0002282	0.0002896
Training Time	02:37 Minutes	

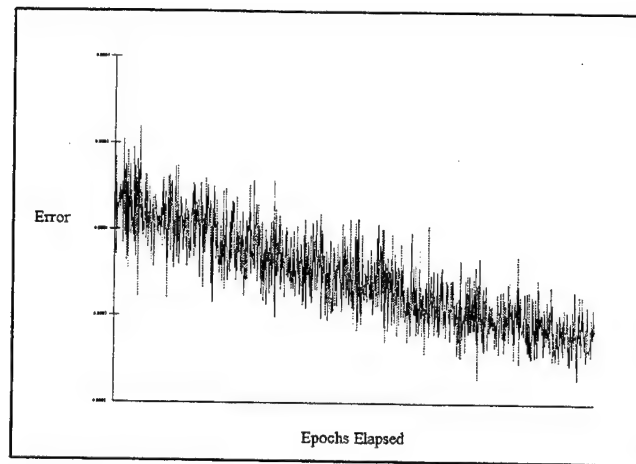


Figure 6. Training Error on Close Network Training Set

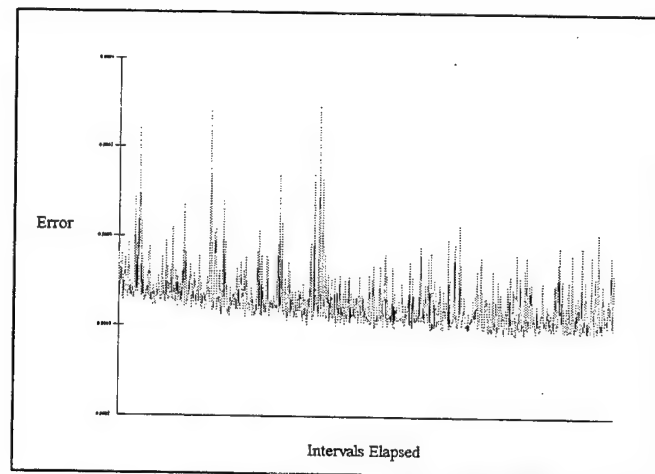


Figure 7. Training Error on Close Network Test Set

The trained network was applied to the 99 out of sample test patterns. The results are shown in Table 2.

Table 2. Close Network Output Statistics

R squared	0.9935
r squared	0.9935
Mean squared error	130.975
Mean absolute error	8.821
Min. absolute error	0.039
Max. absolute error	31.792
Correlation coefficient r	0.9968
Percent within 5%	100
Percent within 5% to 10%	0
Percent within 10% to 20%	0
Percent within 20% to 30%	0
Percent over 30%	0

Figure 8 is a graphical depiction of how the Close Network weighted the contribution of each input to the model.

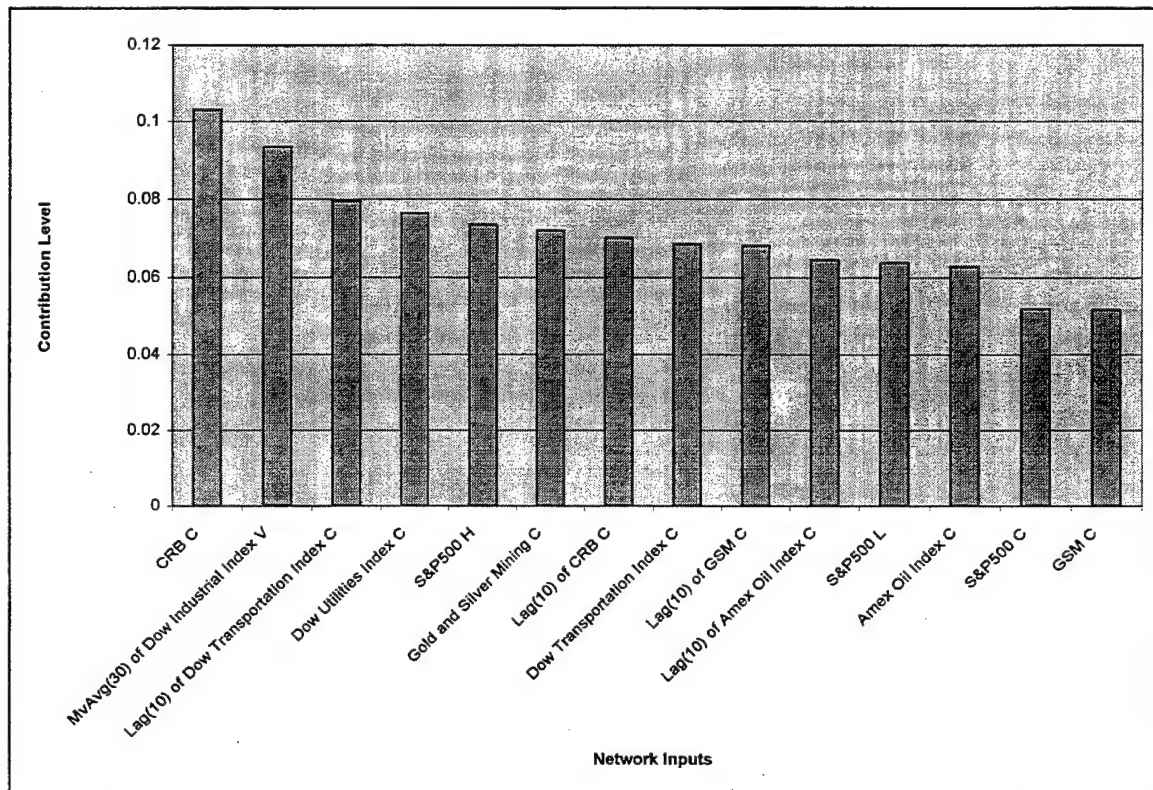


Figure 8. Input Contribution Factors Close Network

Figure 9 is a graphical depiction of the Close Network's forecast versus the actual S&P500 Close.

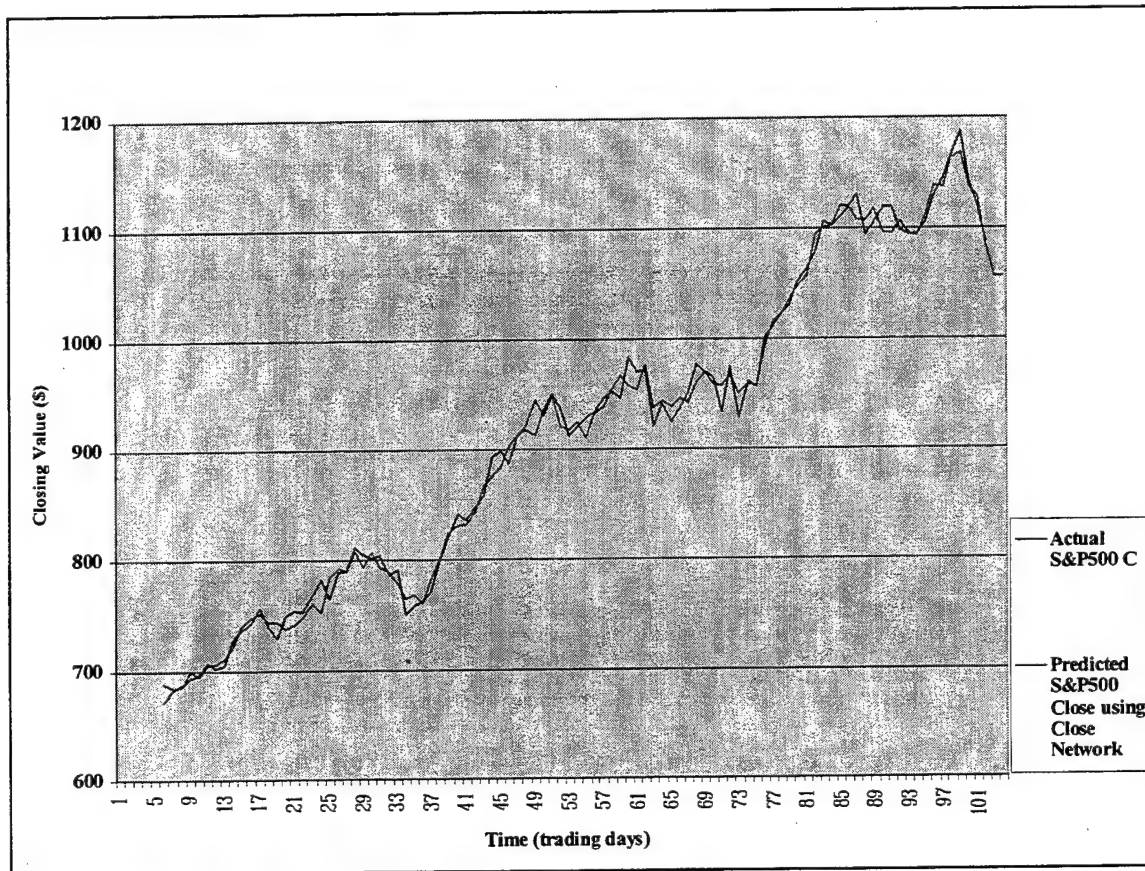


Figure 9. Close Network Actual vs Predicted S&P500 Close

Figure 10 is a graphical depiction of the Close Network's forecast error.

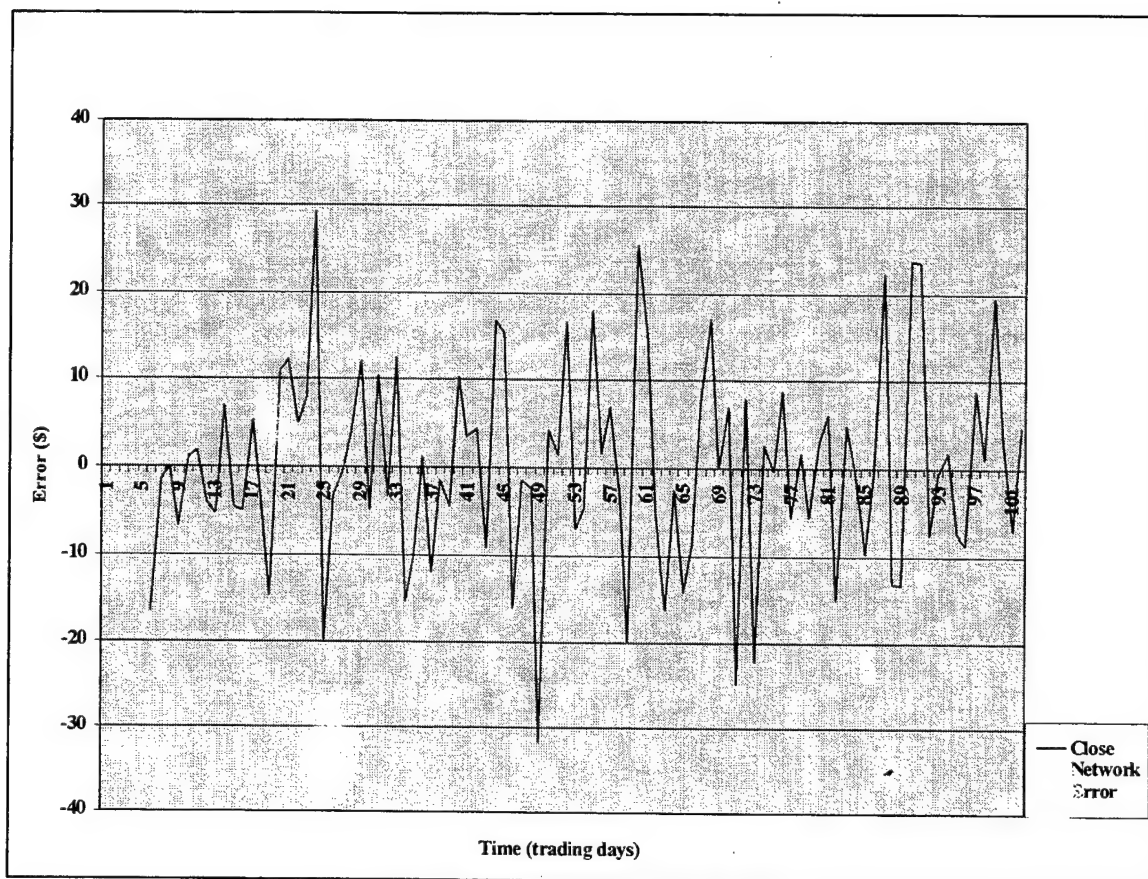


Figure 10. Close Network Error

The Close Network error was input into Excel and the Descriptive Statistics tool was used to generate Table 3.

Table 3. Close Network Descriptive Statistics

Mean	0.175905208
Standard Error	1.167902939
Median	-0.257263184
Standard Deviation	11.50250998
Sample Variance	132.3077358

Table 3 (Continued)

Kurtosis	0.240087928
Skewness	0.050247272
Range	61.15527344
Minimum	-31.79174805
Maximum	29.36352539
Sum	17.06280518
Count	97
Confidence Level(95.0%)	2.318270784

B. PERCENT NETWORK

The following vital data given in Table 4 and Figures 11 and 12 is from the completed training module:

Table 4. Training Module Output Percent Network

	Training Set	Test Set
Learning events	221400	N/A
Learning epochs	572	N/A
Minimum average error	0.0016160	0.0026779
Training Time	01:14 Minutes	

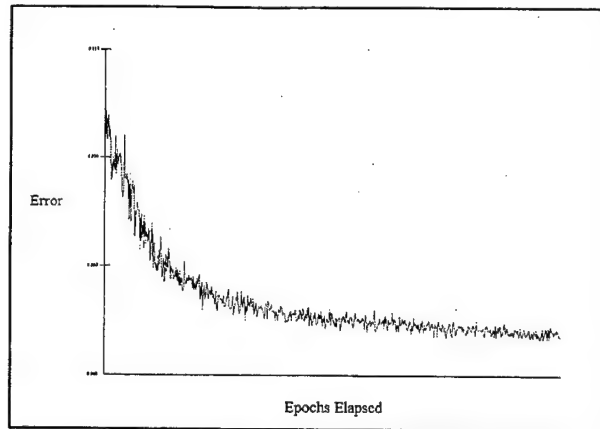


Figure 11. Training Error on Percent Network Training Set

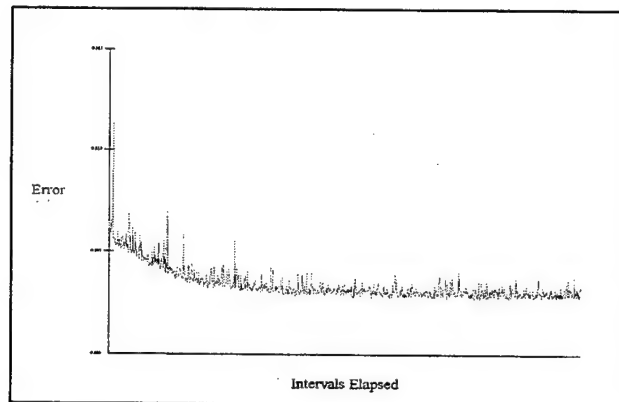


Figure 12. Training Error on Percent Network Test Set

The trained network was applied to the 99 out of sample test patterns. The results are shown in Table 5.

Table 5. Percent Network Output Statistics

R squared	0.7872
r squared	0.7933
Mean squared error	1.449
Mean absolute error	0.919

Table 5 (Continued)

Min. absolute error	0.006
Max. absolute error	4.484
Correlation coefficient r	0.8907
Percent within 5%	10.309
Percent within 5% to 10%	6.186
Percent within 10% to 20%	20.619
Percent within 20% to 30%	11.340
Percent over 30%	51.546

Figure 13 is a graphical depiction of how the Percent Network weighted the contribution of each input to the model.

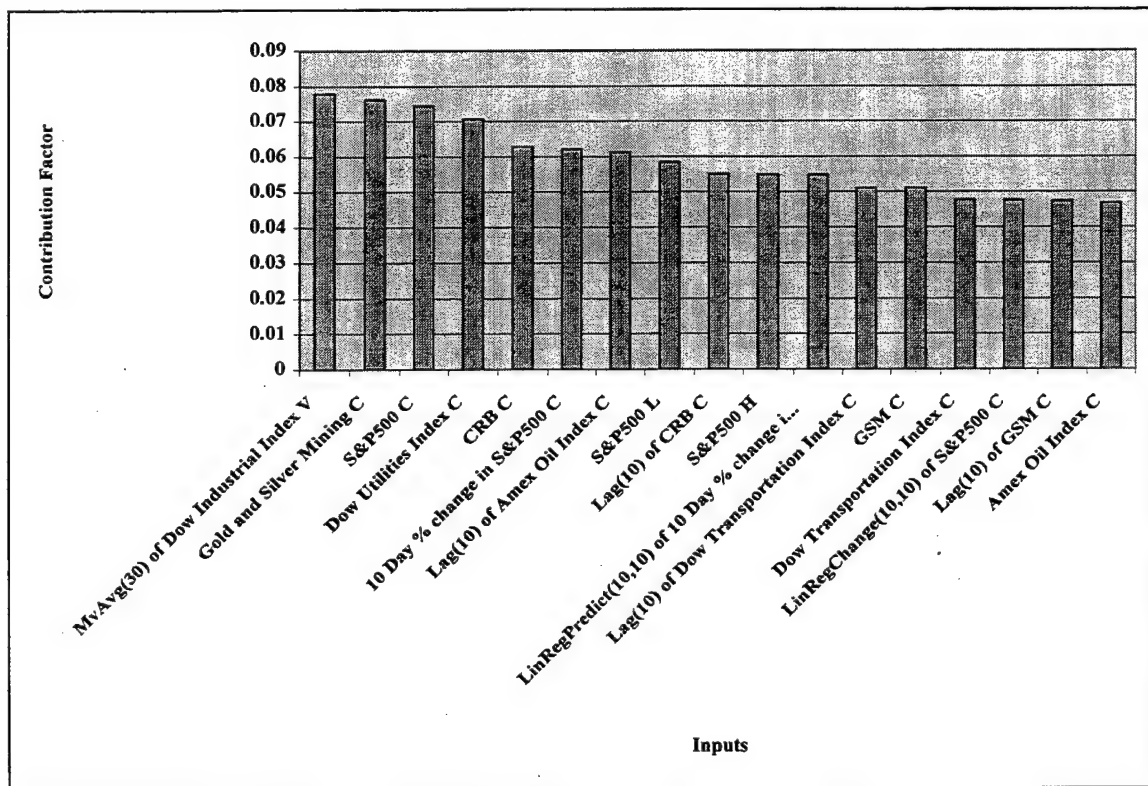


Figure 13. Input Contribution Factors Percent Network

Figure 14 is a graphical depiction of the Percent Network's forecast versus the actual S&P500 Close.

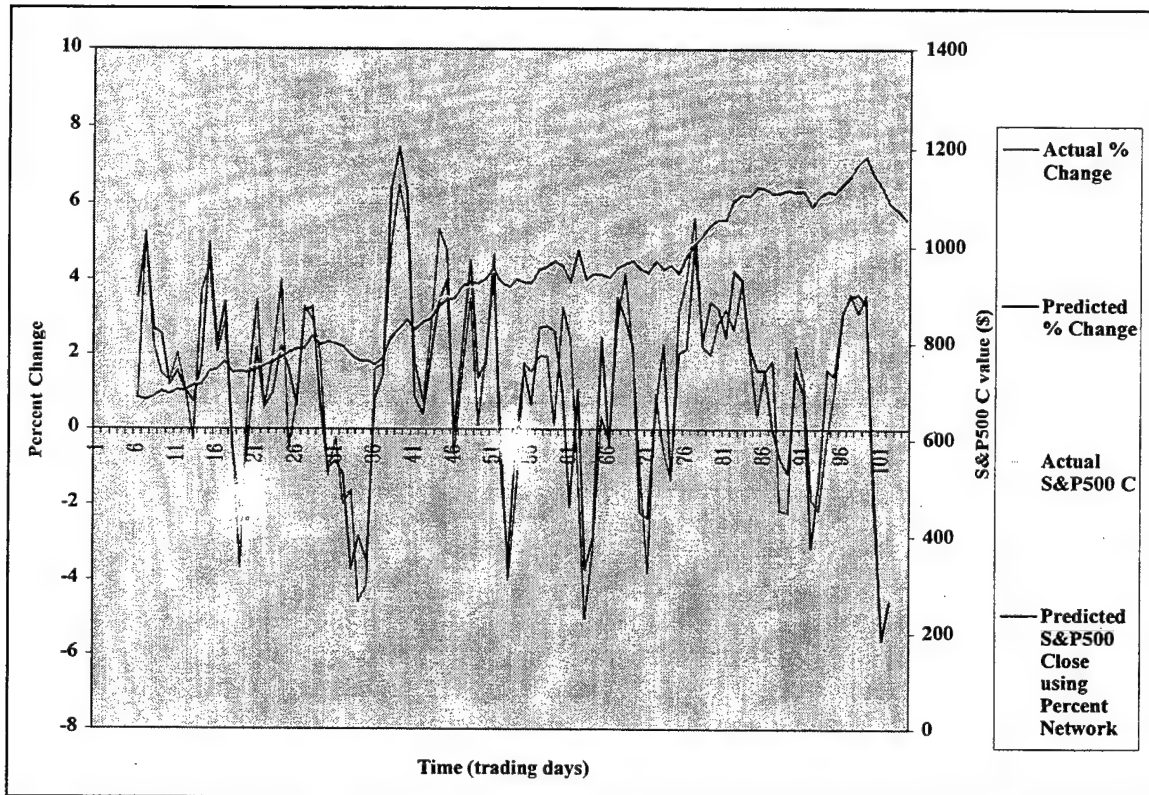


Figure 14. Percent Network Actual vs. Predicted Future 10 Day Percent Change and Closing Price of S&P500

Figure 15 is a graphical depiction of the Percent Network's forecast error.

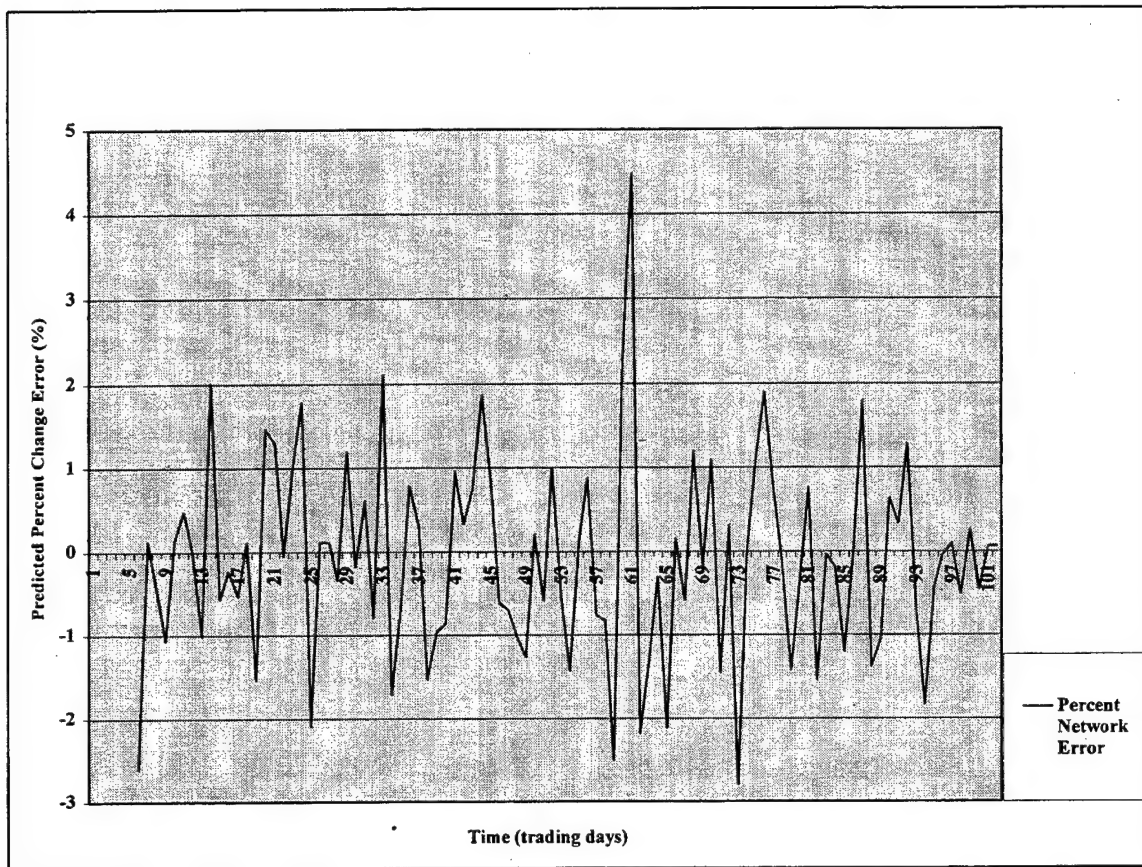


Figure 15. Percent Network Error

The Percent Network error was input into Excel and the Descriptive Statistics tool was used to generate Table 6.

Table 6. Percent Network Descriptive Statistics

Mean	-1.081002006
Standard Error	1.107737379
Median	-0.437339914
Standard Deviation	10.90994792
Sample Variance	119.0269637
Kurtosis	1.792340417
Skewness	0.530272064
Range	68.62393946
Minimum	-26.16139546
Maximum	42.462544
Sum	-104.8571946
Count	97
Confidence Level(95.0%)	2.198842999

Figure 16 is a graphical depiction of the both the Close and Percent Network's forecast versus the actual S&P500 Close.

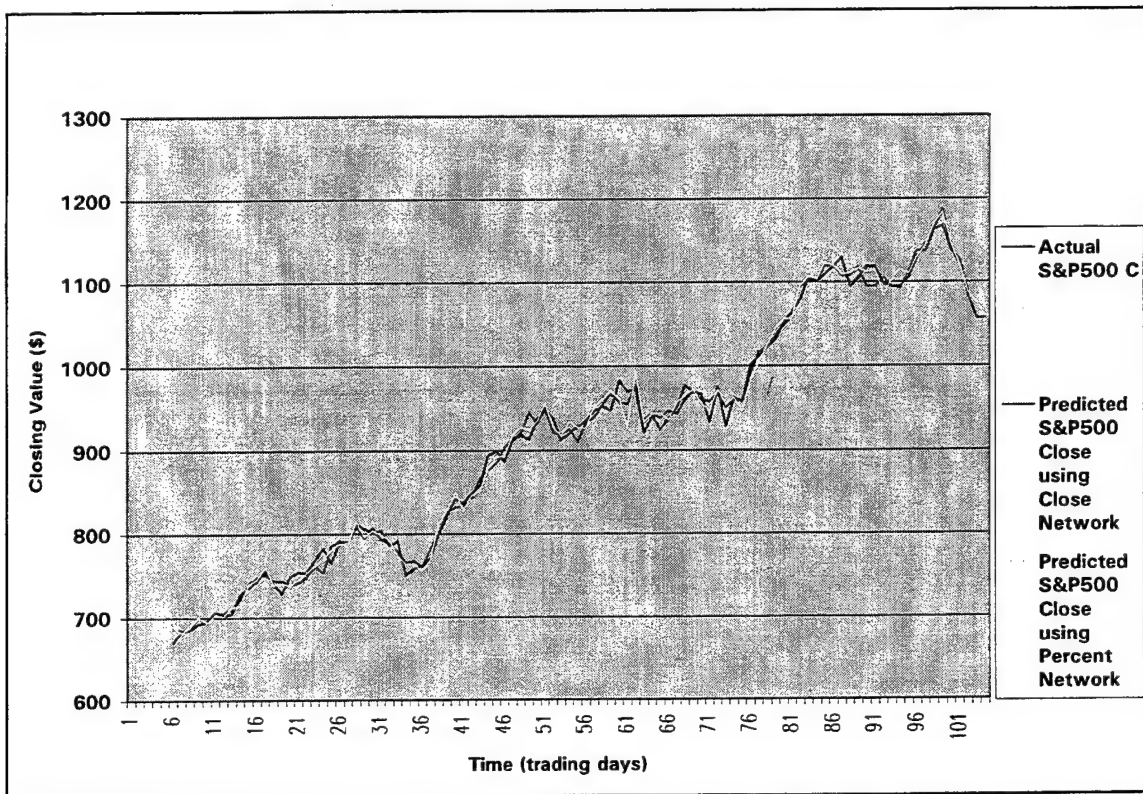


Figure 16. All Network Models Actual vs. Predicted S&P500 C

Figure 17 is a graphical depiction of both the Percent Network and Close Network forecast error.

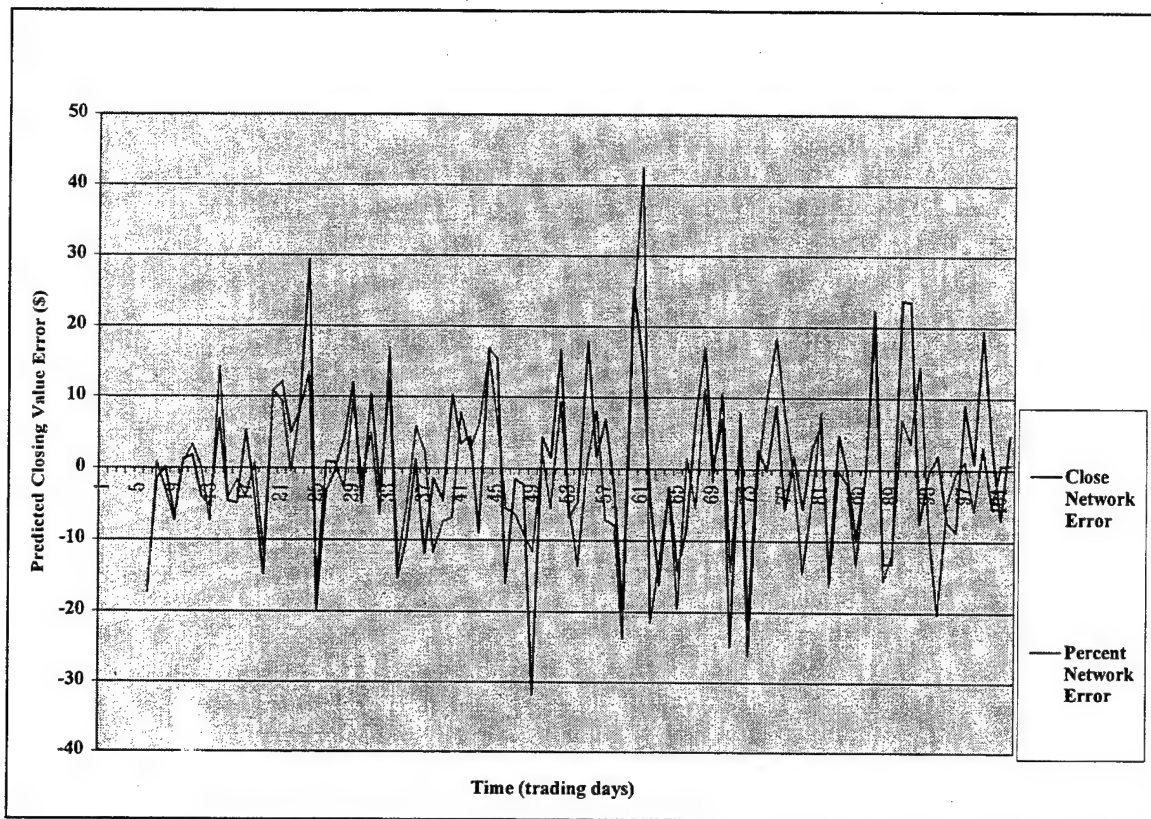


Figure 17. S&P500 Closing Value Prediction Neural Network Model Error Chart

C. MULTIPLE LINEAR REGRESSION MODEL

Multiple linear regression analysis was performed using the Data Analysis tool within Excel. The output of this process for the final model is presented in Table 7.

Table 7. Regression Statistics

Multiple R	0.983432302
R Square	0.967139093
Adjusted R Square	0.96679536
Standard Error	25.81440036
Observations	484

Table 7 (Continued)

ANOVA				
	Df	SS	MS	F
Regression	5	9374786.243	1874957.249	2813.63195
Residual	478	318531.2012	666.3832661	
Total	483	9693317.444		
	Coefficients	Standard Error	t Stat	P-value
Intercept	1523.248565	96.30217982	15.81738407	1.25634E-45
Dow Transportation Index C	0.173215277	0.008063272	21.48200959	3.79263E-72
Dow Utilities Index C	-0.512956407	0.157214185	-3.262787046	0.00118219
Amex Oil Index C	1.455027957	0.117440864	12.38945207	9.13351E-31
CRB C	-5.159128047	0.28529288	-18.08362007	4.48872E-56
Lag(10) of Amex Oil Index C	-0.902950239	0.106686284	-8.463601907	3.19588E-16

Excel was also used to calculate the Variance Inflationary Factor (VIF). The results are given below in Table 8.

Table 8. VIF for Regression Inputs

Dependant Variable	R Square	Variance Inflationary Factor (VIF)
Dow Transportation Index C	0.918384	6.39
Dow Utilities Index C	0.911393	5.90
Amex Oil Index C	0.942674	8.98
CRB C	0.855026	3.72
Lag(10) of Amex Oil Index C	0.936325	8.11

The Normal probability plot, Line Fit Plots, and Residual Plots for the Multiple Regression Model are provided in Appendix A. Figure 18 is a graphical depiction of the Multiple Regression Model's forecast versus the actual S&P500 Close.

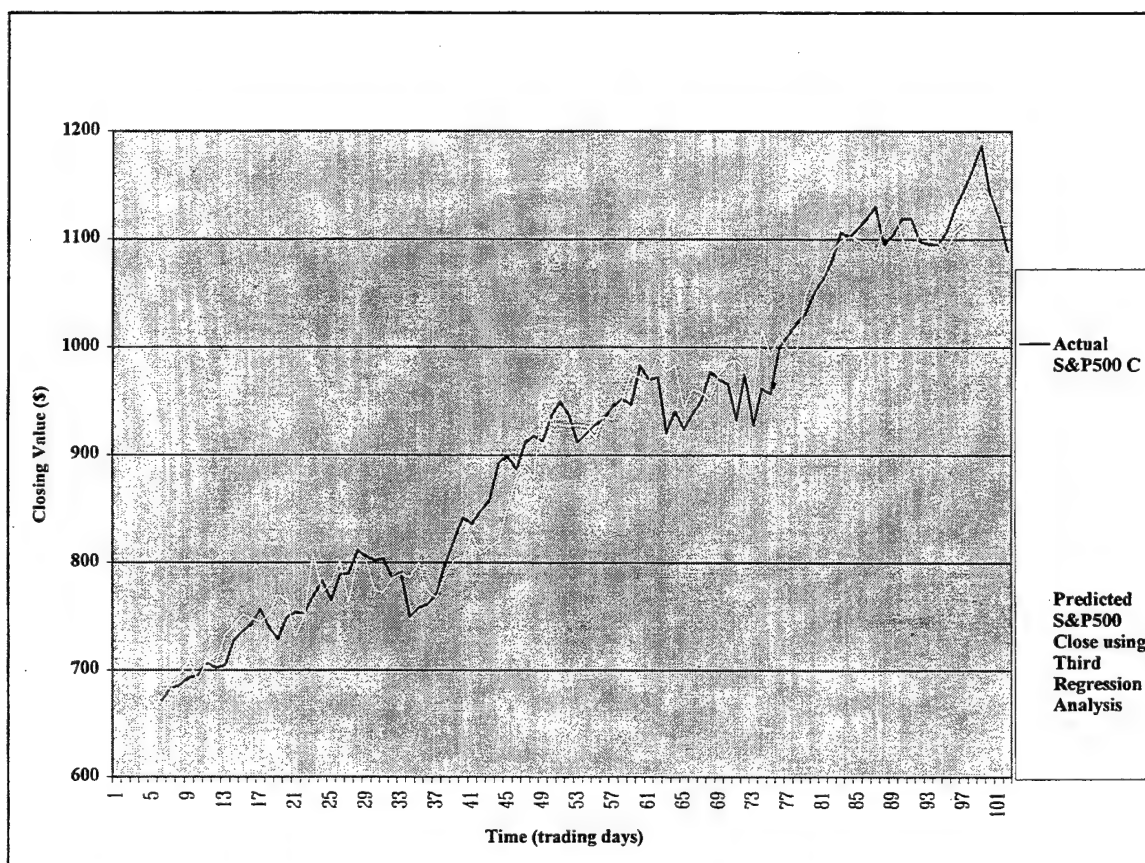


Figure 18. Regression Model Predicted vs. Actual S&P500 Close

Figure 19 is a graphical depiction of the Multiple Regression Model's forecast error.

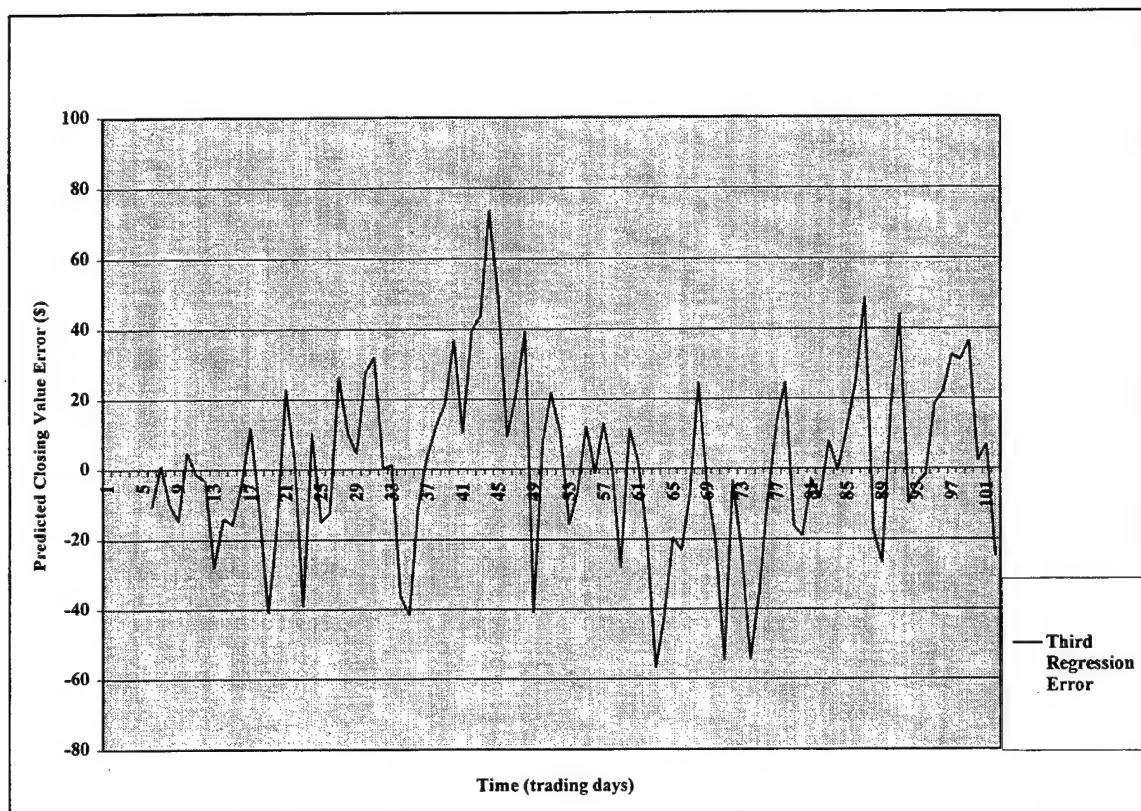


Figure 19. Regression Model Error Chart

The Regression Model error was input into Excel and the Descriptive Statistics tool was used to generate Table 9.

Table 9. Regression Model Descriptive Statistics

Mean	0.291446988
Standard Error	2.533807729
Median	0.001038339
Standard Deviation	24.95511202
Sample Variance	622.7576161

Table 9 (Continued)

Kurtosis	0.138974197
Skewness	0.071861212
Range	130.4937839
Minimum	-56.84815562
Maximum	73.64562832
Sum	28.27035783
Count	97
Confidence Level(95.0%)	5.029572436

Figure 20 is a graphical depiction of all mode forecasts versus the actual S&P500 Close.

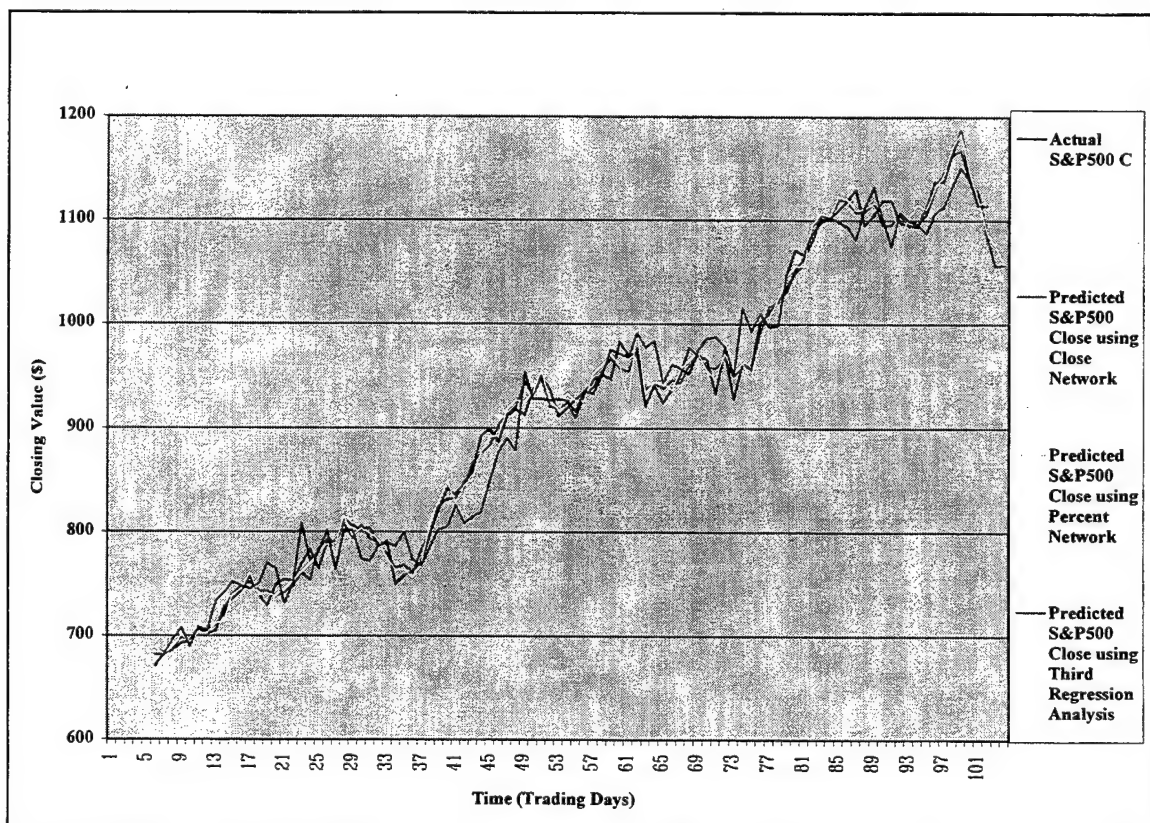


Figure 20. Actual vs. All Model Predictions of S&P500 Close Ten Days Into Future

Figure 21 is a graphical depiction of all models' forecast error.

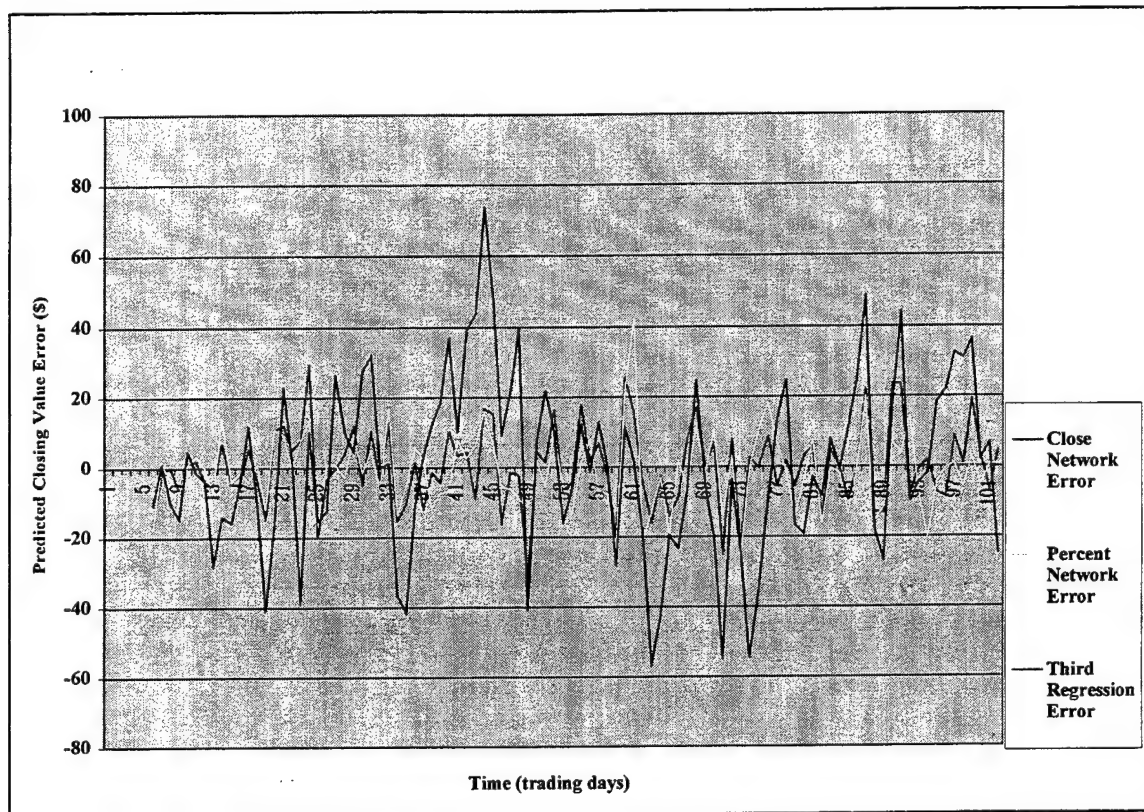


Figure 21. All Model Error Chart

D. CONDITIONAL PROBABILITY

Historical data was calculated for the number of times the market rose and fell during the period from March 3rd, 1991 to August 18th, 1998. The data in Table 10 applies:

Table 10. Historical Market Data

INITIAL KNOWLEDGE	
Total Days the S&P500 Rose	1173
Total Days the S&P500 Fell	696
Total Days	1869

The accuracy of each model was then calculated. The data in Tables 11, 12, and 13 apply:

Table 11. Close Network Accuracy

		Accuracy of Model		
		States of Nature		
		Rise	Fall	Total
Close Network Model	Rise	62	6	68
	Fall	6	23	29
	Total	68	29	97

Table 12. Percent Network Accuracy

		Accuracy of Model		
		States of Nature		
		Rise	Fall	Total
Percent Network Model	Rise	65	3	68
	Fall	5	24	29
	Total	70	27	97

Table 13. Multiple Regression Accuracy

		Accuracy of Model		
		States of Nature		
		Rise	Fall	Total
Multiple Regression Model	Rise	54	14	68
	Fall	15	14	29
	Total	69	28	97

The model accuracy was then adjusted using the environmental data. The data in Tables 14, 15, and 16 apply:

Table 14. Close Network Environmental Adjustment

		Adjustment		
		States of Nature		
		Rise	Fall	Total
Close Network Model	Rise	1069.5	144	1213.5
	Fall	103.5	552	655.5
	Total	1173	696	1869

Table 15. Percent Network Environmental Adjustment

		Adjustment		
		States of Nature		
		Rise	Fall	Total
Percent Network Model	Rise	1089.21	77.3333	1166.54
	Fall	83.7857	618.666	702.452
	Total	1173	696	1869

Table 16. Multiple Regression Environmental Adjustment

		Adjustment		
		States of Nature		
		Rise	Fall	Total
Multiple Regression Model	Rise	918	348	1266
	Fall	255	348	603
	Total	1173	696	1869

Finally the prediction accuracy probability was calculated for each model. The data in Tables 17, 18, and 19 apply.

Table 17. Close Network Accuracy Probability

	Prediction	Probability
Close Network Model	Market Rise	88.1335
	Market Fall	84.21053

Table 18. Percent Network Accuracy Probability

	Prediction	Probability
Percent Network Model	Market Rise	93.37075
	Market Fall	88.0724

Table 19. Multiple Regression Accuracy Probability

	Prediction	Probability
Multiple Regression Model	Market Rise	72.51185
	Market Fall	57.71144

V. DATA ANALYSIS AND INTERPRETATION

Model accuracy can be compared across the three models for the following characteristics:

- Coefficient of Multiple Determination.
- Combined Actual vs. Predicted S&P500 Close Chart.
- Error statistics, specifically the mean and standard deviation.
- All Model Error Chart.
- Conditional Probabilities.

Table 20 compares the ranked Coefficients of Multiple Determination for each model:

Table 20. Coefficients of Multiple Determination

Model	R square
Close Network	.9935
Regression Model	.9671
Percent Network	.7872

The R square value represents the proportion of variation in the dependant variable that is explained by the independent variables. The better the model explains variation in the dependant variable, the higher the R square value. Without further comparison, the Close Network best explains variation in the dependant variable, followed by the Regression Model.

In examining Figure 20, Actual vs. All Model Predictions of S&P500 Close Ten Days into the Future, it is relatively easy to visually verify that the two network models perform better than the regression model. This differs from the model ranking due to R square values. Both network models predict the closing value relatively accurately. However, it is difficult visually to determine which of these is more accurate.

In Table 21, the ranked error statistics are provided for comparison. These statistics are all based on closing price error. The Percent Network error was converted to closing price error, so it could be compared to the other two models.

Table 21. Model Error Statistics

Model	Mean	Standard Deviation
Percent Network	-1.08	10.9
Close Network	.1759	11.50
Regression Model	.2914	24.95

Ideally, the mean error would be zero and the standard deviation would be as small as possible. All of the models' means are relatively close to zero. However, the breakout occurs with standard deviation. Generally it can be said that there is a 95.44 percent probability of the error values falling within two standard deviations of the mean. Therefore, the larger the standard deviation the greater the range of error. Although it has a lower coefficient of determination, the Percent Network is more accurate. This tends to support the theory that predicting a percent change is more accurate than predicting the closing value.

Visually examining Figure 21, All Model Error Chart, clearly shows the Regression Model has the greatest error variation. Differentiation between the two network models is difficult, suggesting that the error statistics should provide the true ranking.

The final ranking test is the conditional accuracy probabilities for each model. Table 22 summarizes and ranks these probabilities for easy comparison.

Table 22. Conditional Probabilities

Model	Probability of Accurately Predicting	
	Market Rise	Market Fall
Percent Network	93.37	88.07
Close Network	88.13	84.21
Regression Model	72.5	57.71

Clearly, the prediction probabilities support the error statistics rankings and verify that the Percent Network is the most accurate predictor of the S&P500 closing price.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

It is clear from the model accuracy probabilities that neural networks can accurately predict financial markets if given the proper data upon which to train. When compared to regression analysis, neural networks are a better tool for the investor for the following reasons:

- When using multiple linear regression, the governing regression assumptions must be true. The linearity assumption itself may not hold in many cases. Neural networks can model both linear and curvilinear systems.
- When using multiple linear regression analysis, the investor must have a deep understanding of statistics to ensure only the necessary independent variables are used. This is true only to a limited degree with neural networks and can be mitigated by the systematic removal method described by Jurik in *Virtual Trading*.
- For the models studied in this research, neural networks are significantly more accurate than multiple linear regression analysis.

However, the difficulty in identifying good raw data, preprocessing this data, training a network and repeating this process until a good model is developed should not be discounted. The individual investor should be warned that model accuracy is difficult to obtain and can take months or years of investigation.

B. RECOMMENDATIONS

For individual investors to successfully use neural networks to predict financial markets, they must undertake the following:

- As a minimum, read a basic text on neural networks to understand neural network theory and its limitations.
- Understand basic statistical measures and probability.
- Become extremely proficient with a spreadsheet software program so that data can be preprocessed efficiently.
- Study financial market theory so that input data to the neural net is not chosen inappropriately or at random.
- Obtain a neural network software program and test and evaluate many neural networks. Practice by varying inputs and studying the effects on outputs.
- Once a network is developed, do not blindly follow its advice. As Gately (1996) states, try to verify it with other indicators before taking action. In other words use multiple indicators. This could include using multiple networks incorporating different inputs to predict the same output.

Only by meticulously following these recommendations will individual investors improve their potential profits by using neural networks.

Although outside the scope of this research, an important question for any forecasting model is how does the model stand up in an actual trading scenario? Each of these models should be historically tested as if being used for actual trading. Given the same initial capital investment, what is the return for each of these models? What is the maximum drawdown or capital loss for each of these models? If a model has a high return is it due to many steady small gains or is it due to one huge gain surrounded by many losses? These issues revolve around profit and not model accuracy. This raises the question of whether it is important to design a network and train it to maximize profit vs.

forecasting accuracy? These are all critical questions that take the forecasting model from academic research into the actual world of trading.

APPENDIX A. THIRD REGRESSION PARTIAL OUTPUT AND VITAL PLOTS

THIRD REGRESSION SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.983432302
R Square	0.967139093
Adjusted R Square	0.96679536
Standard Error	25.81440036
Observations	484

ANOVA				
	df	SS	MS	F
Regression	5	9374786.243	1874957.249	2813.63195
Residual	478	318531.2012	666.3832661	
Total	483	9693317.444		

	Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%
Intercept	1523.248565	96.30217982	15.81738407	1.25634E-45	1334.020505	1712.476625	1334.020505	1712.476625
Dow Transportation Index C	0.17215277	0.008063272	21.48200959	3.79263E-72	0.157371428	0.189059127	0.157371428	0.189059127
Dow Utilities Index C	-0.512956407	0.157214185	-3.262787046	0.00118219	-0.821872937	-0.204039877	-0.821872937	-0.204039877
Amex Oil Index C	1.455027957	0.117440864	12.38945207	9.13351E-31	1.224263639	1.685792274	1.224263639	1.685792274
CRB C	-5.159128047	0.28529288	-18.08362007	4.48872E-56	-5.7197116	-4.598544494	-5.7197116	-4.598544494
Lag(10) of Amex Oil Index C	-0.902950239	0.106866284	-8.463601907	3.19588E-16	-1.112582446	-0.693318031	-1.112582446	-0.693318031

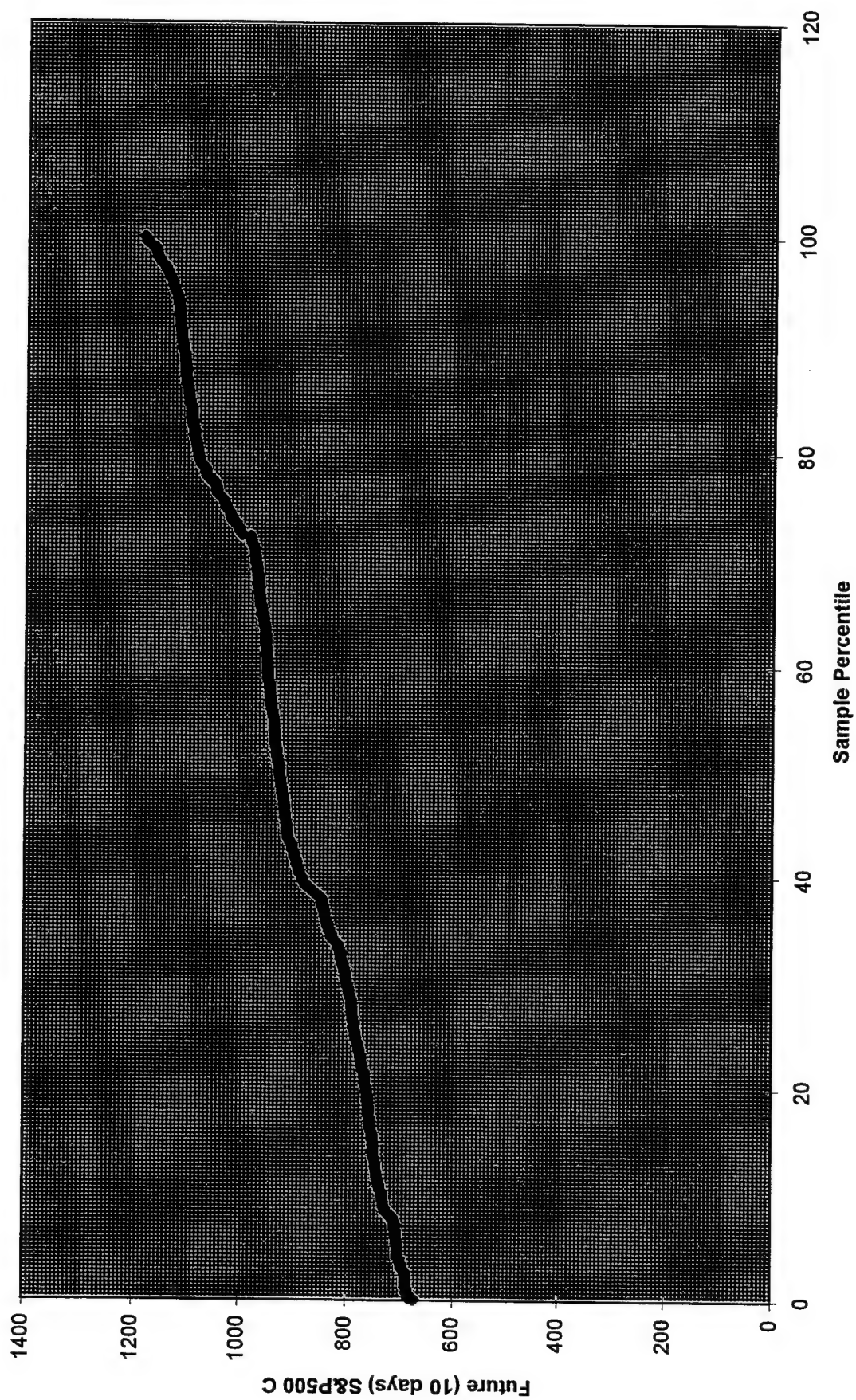
RESIDUAL OUTPUT

Observation	Predicted Future (10 days) Std 500 C	Residuals	Standard Residuals
1	681.823017	-10.82301703	-0.421449864
2	669.8340231	11.16597689	0.43480477
3	665.77933	18.22066998	0.70951555
4	673.7919767	9.208023315	0.358561772
5	680.3835598	0.416440197	0.016216242
6	682.0541838	0.945816229	0.036830222
7	678.2896603	8.710039653	0.339170216
8	686.0147662	-0.014766183	-0.000574997
9	684.9169039	1.083096139	0.042175922
10	695.8797107	-9.879710731	-0.384717378
11	696.0823088	-10.08230885	-0.392606579
12	708.112211	-22.11221097	-0.861052724
13	721.0778502	-34.07785017	-1.326996462
14	706.9560834	-17.95608316	-0.699212508
15	700.3146103	-6.314610297	-0.245891847
16	707.8388262	-14.83882624	-0.577826061

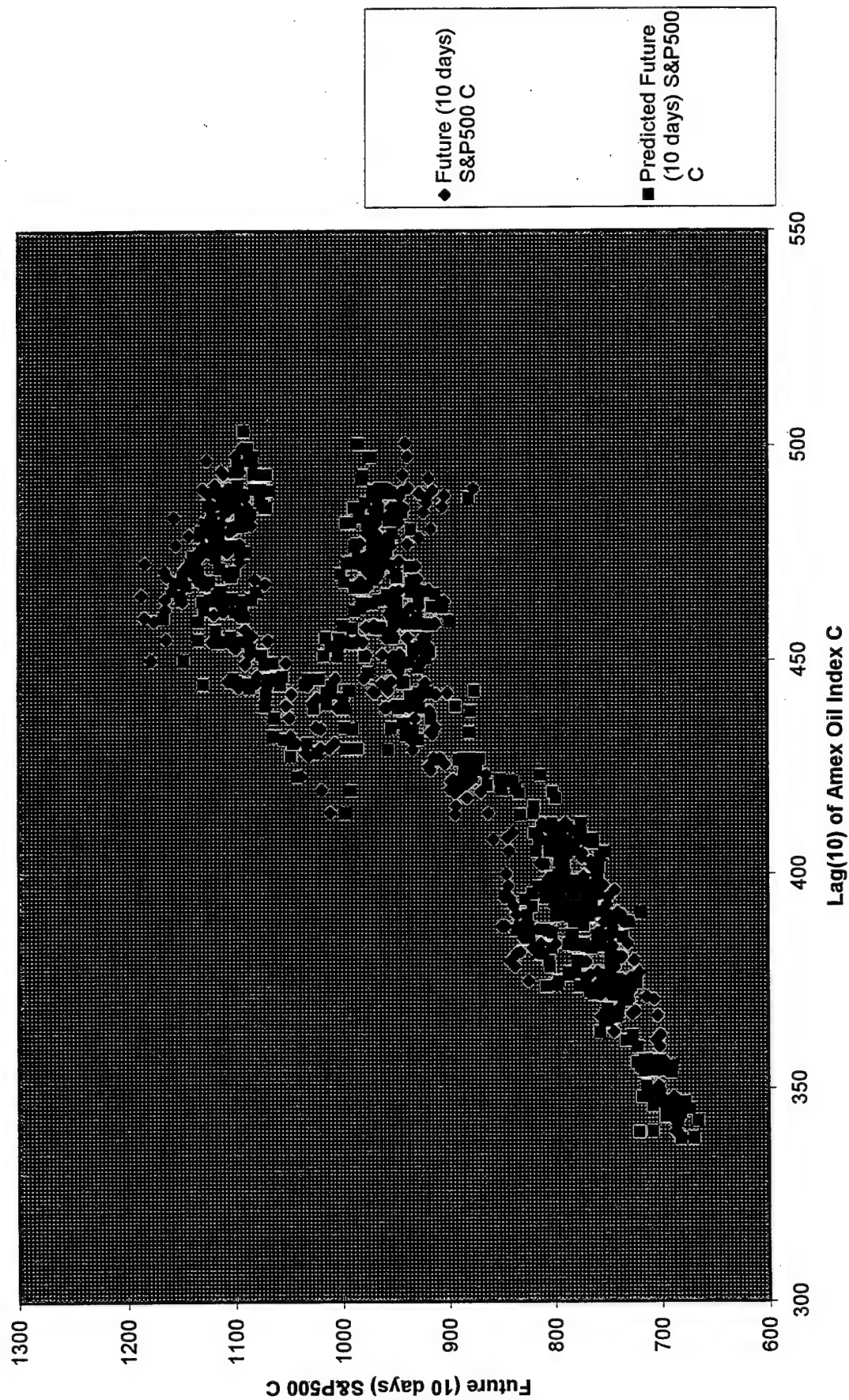
PROBABILITY OUTPUT

Percentile	Future (10 days) Std 500 C
0.103035785	671
0.309917355	681
0.516528926	681
0.723140496	683
0.929752066	683
1.136363636	684
1.342975207	686
1.549586777	686
1.756198347	686
1.962809917	686
2.169421488	686
2.376033058	687
2.582644628	687
2.789256198	689
2.995867769	693
3.202479339	694

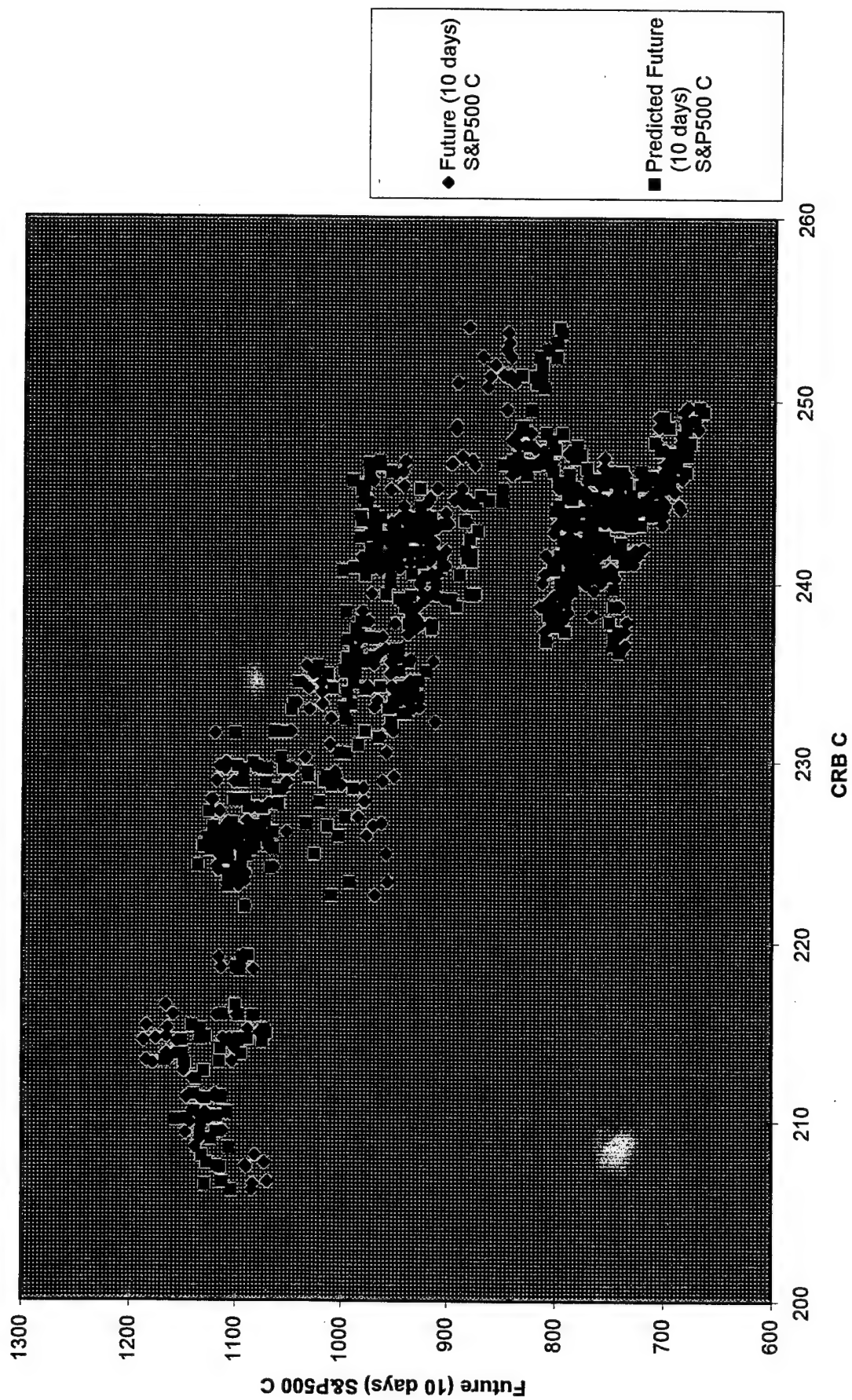
Normal Probability Plot



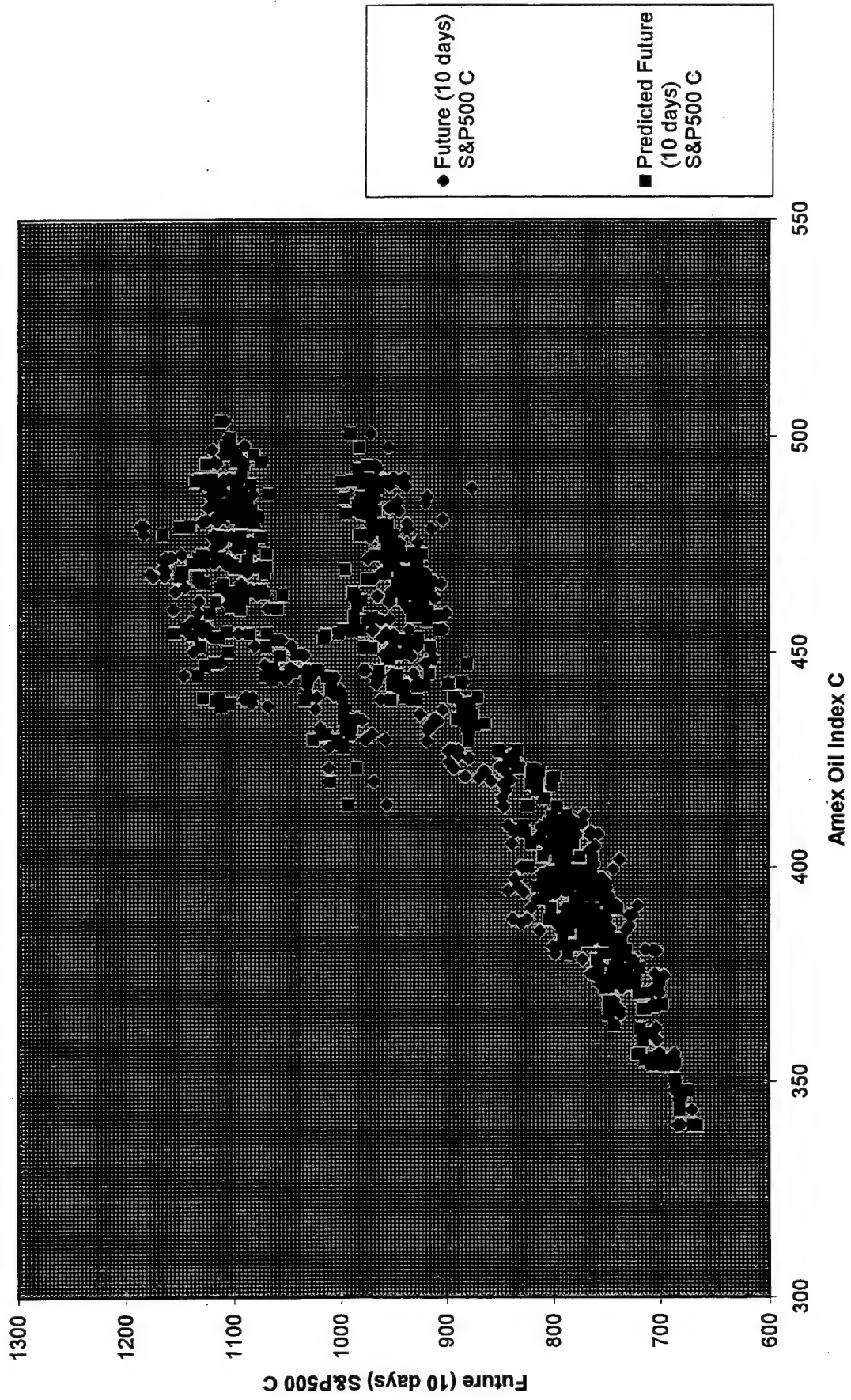
Lag(10) of Amex Oil Index C Line Fit Plot



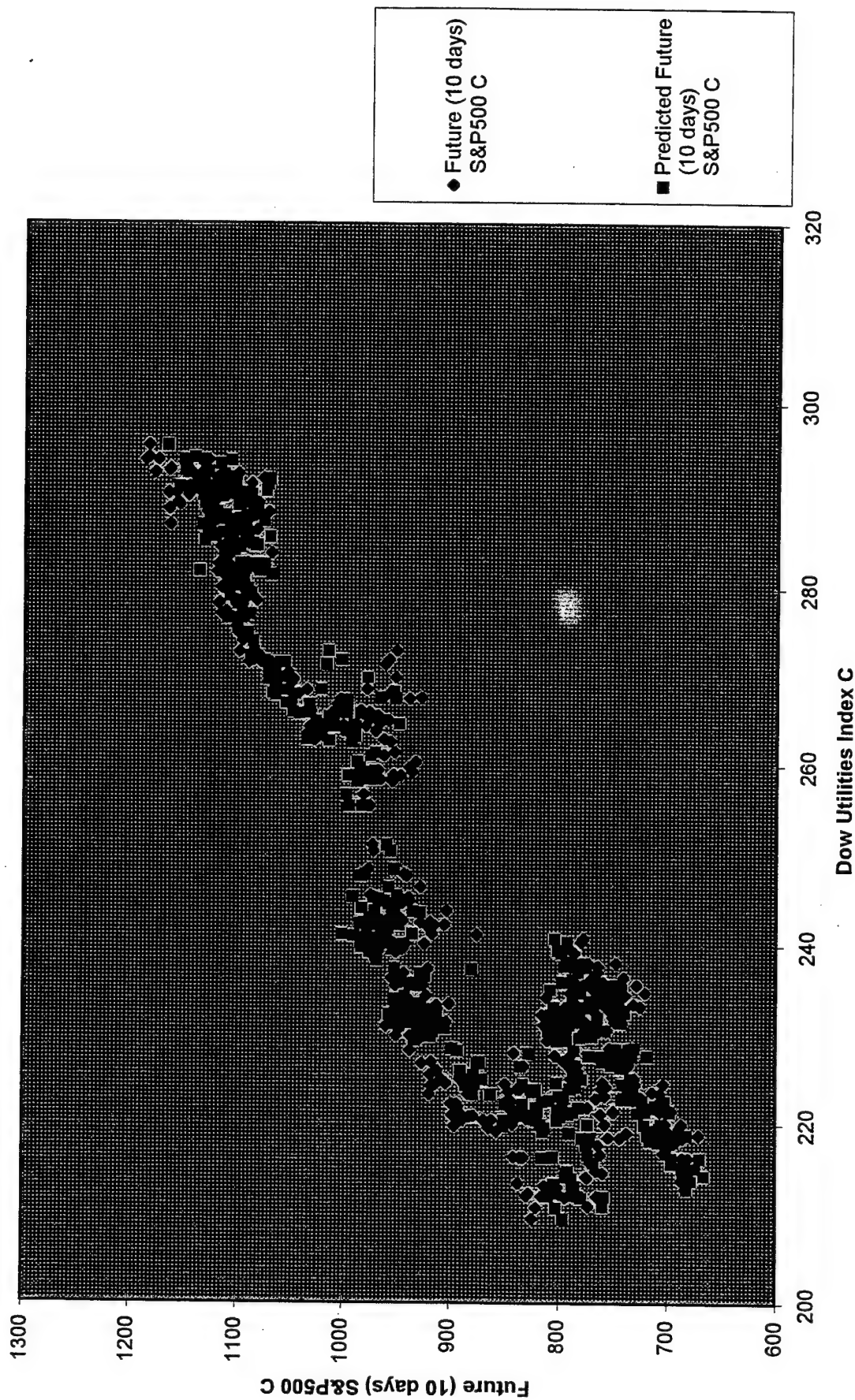
CRB C Line Fit Plot



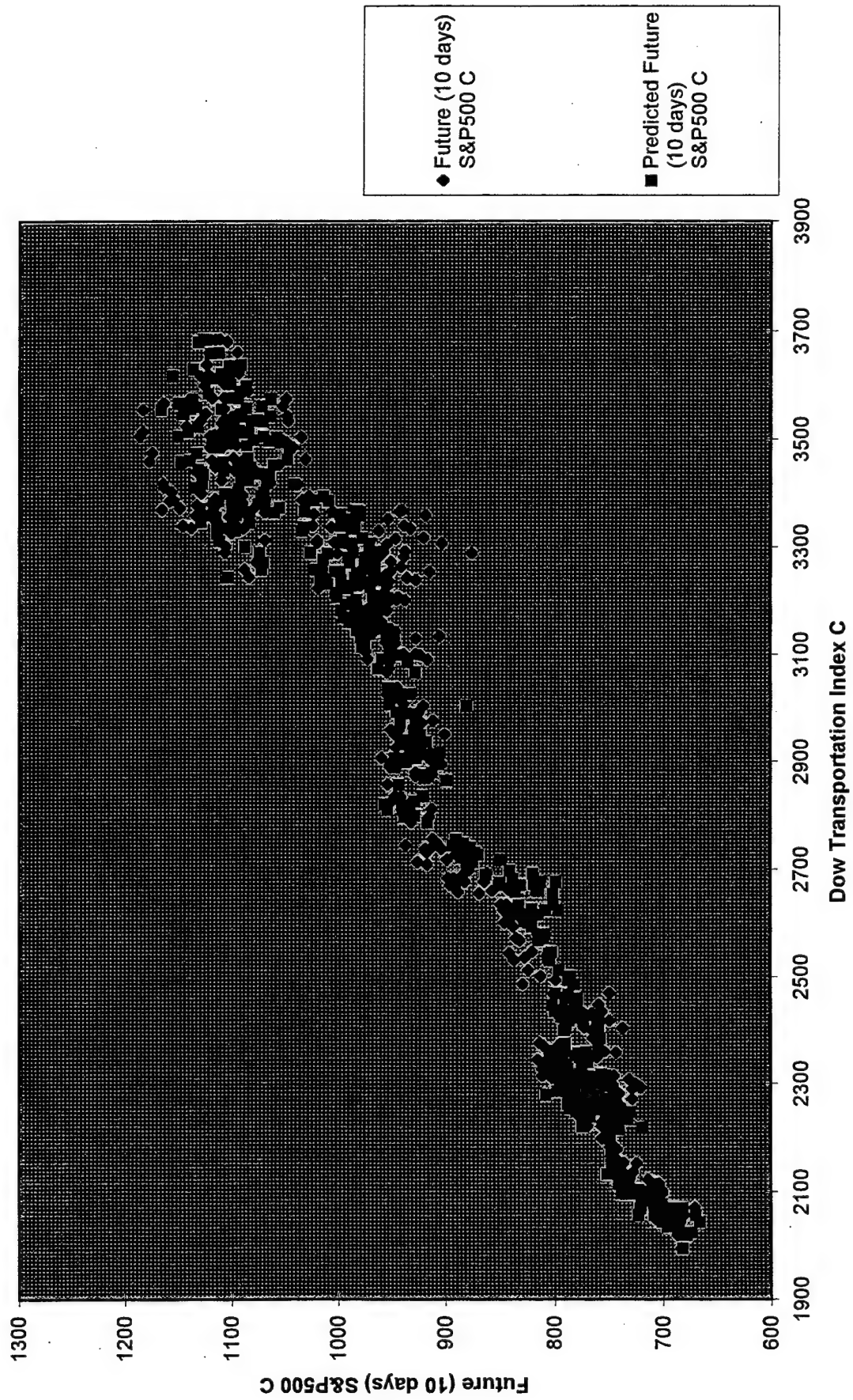
Amex Oil Index C Line Fit Plot



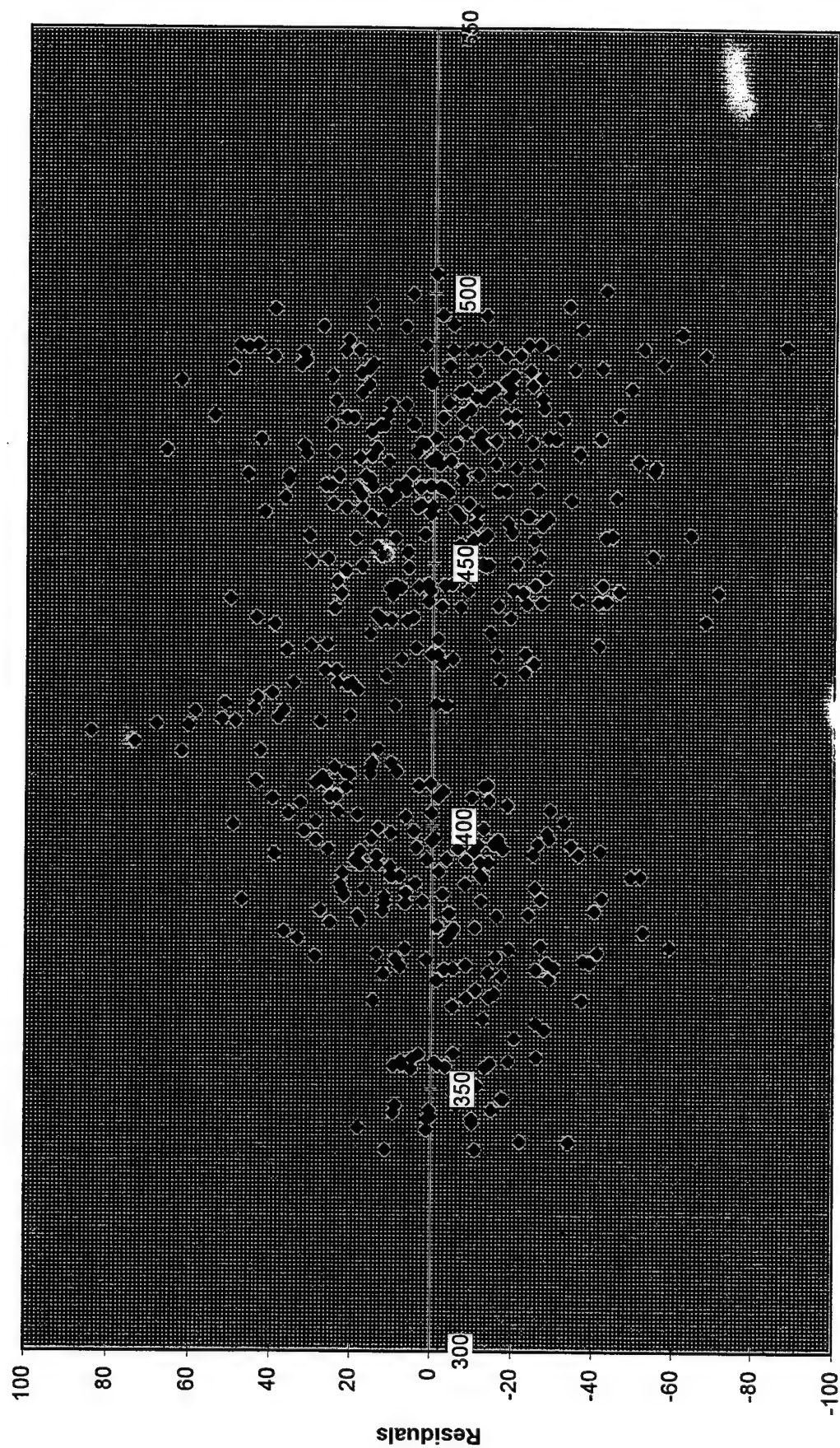
Dow Utilities Index C Line Fit Plot



Dow Transportation Index C Line Fit Plot

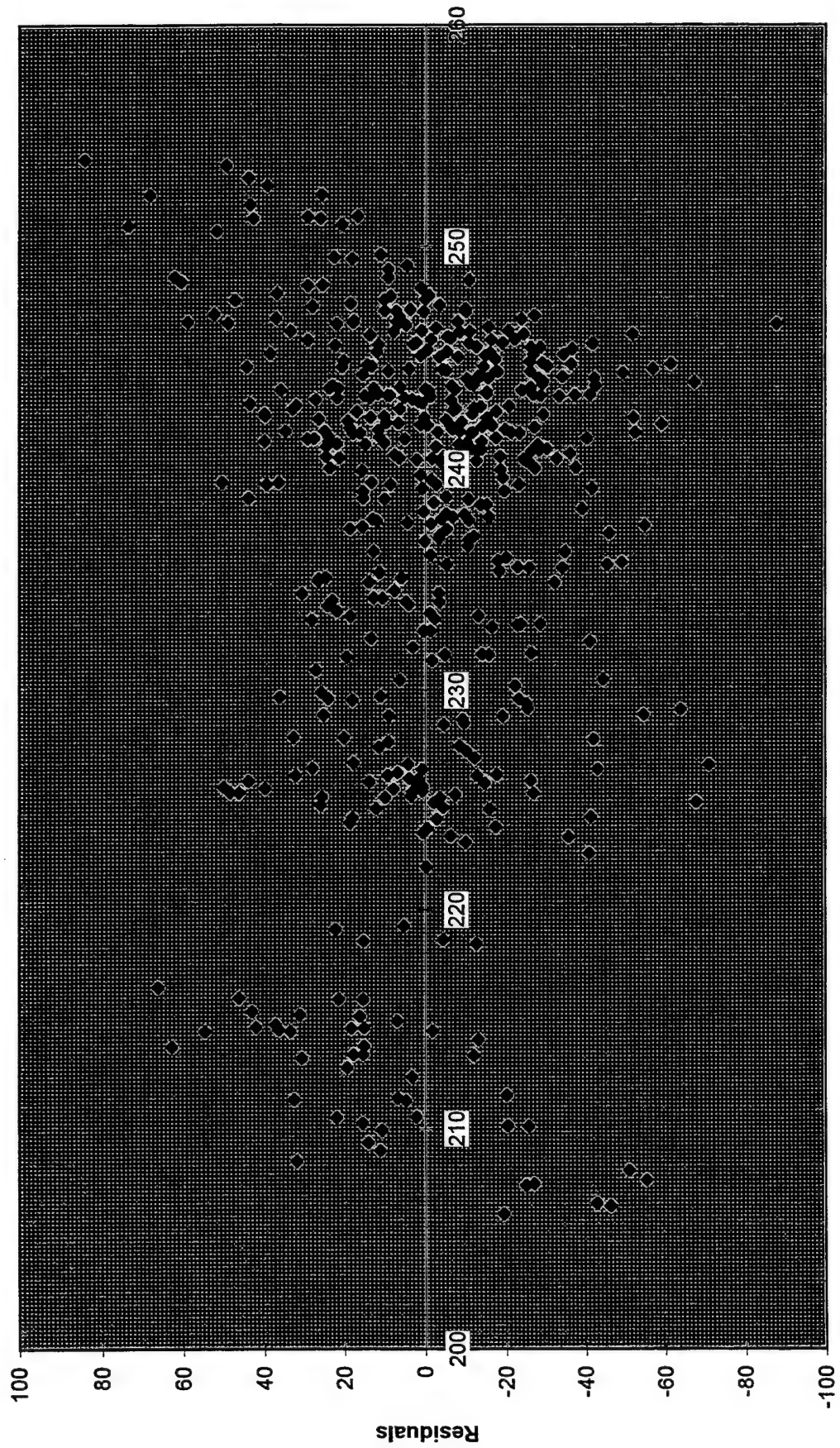


Lag(10) of Amex Oil Index C Residual Plot



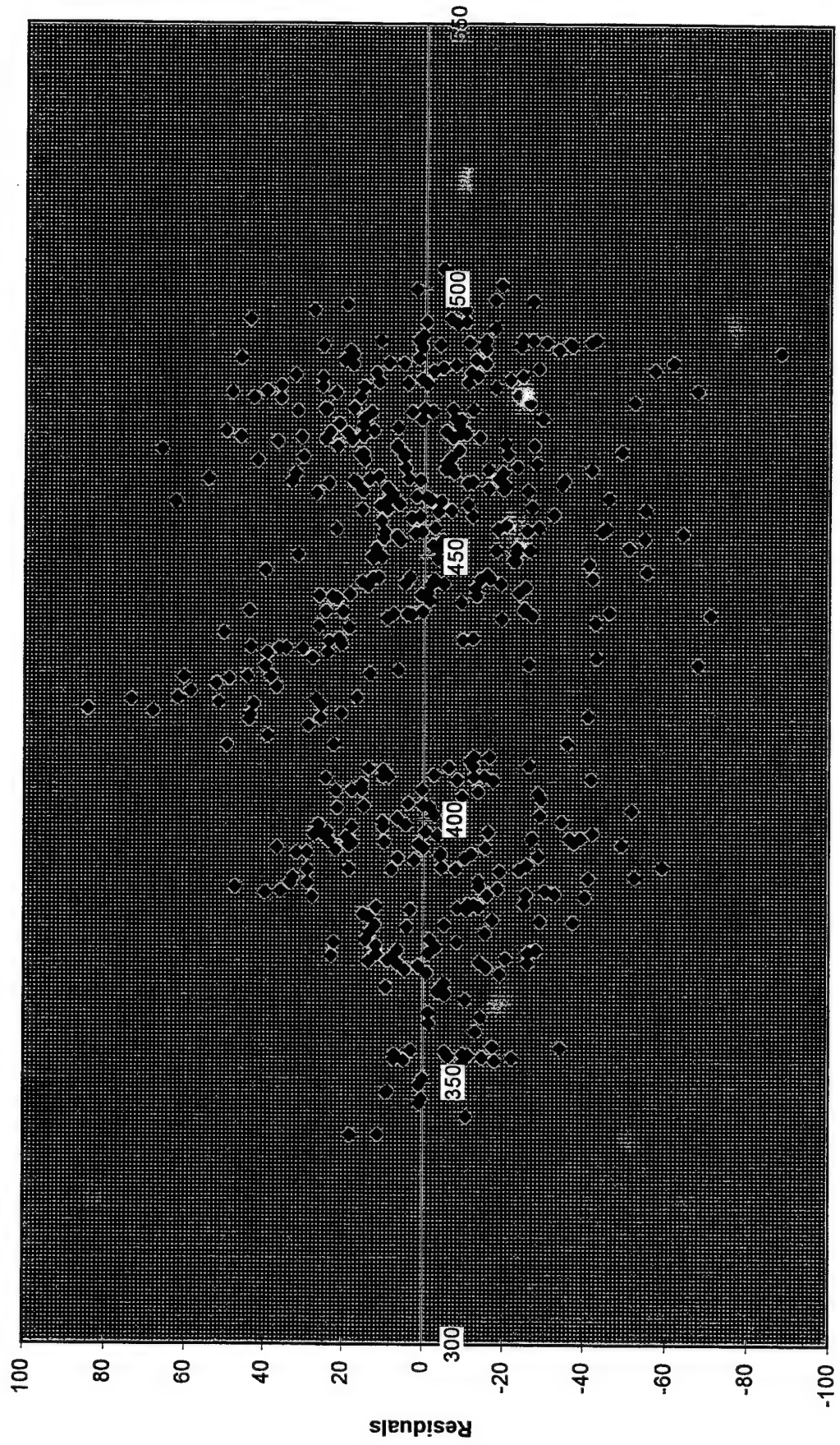
Lag(10) of Amex Oil Index C

CRB C Residual Plot

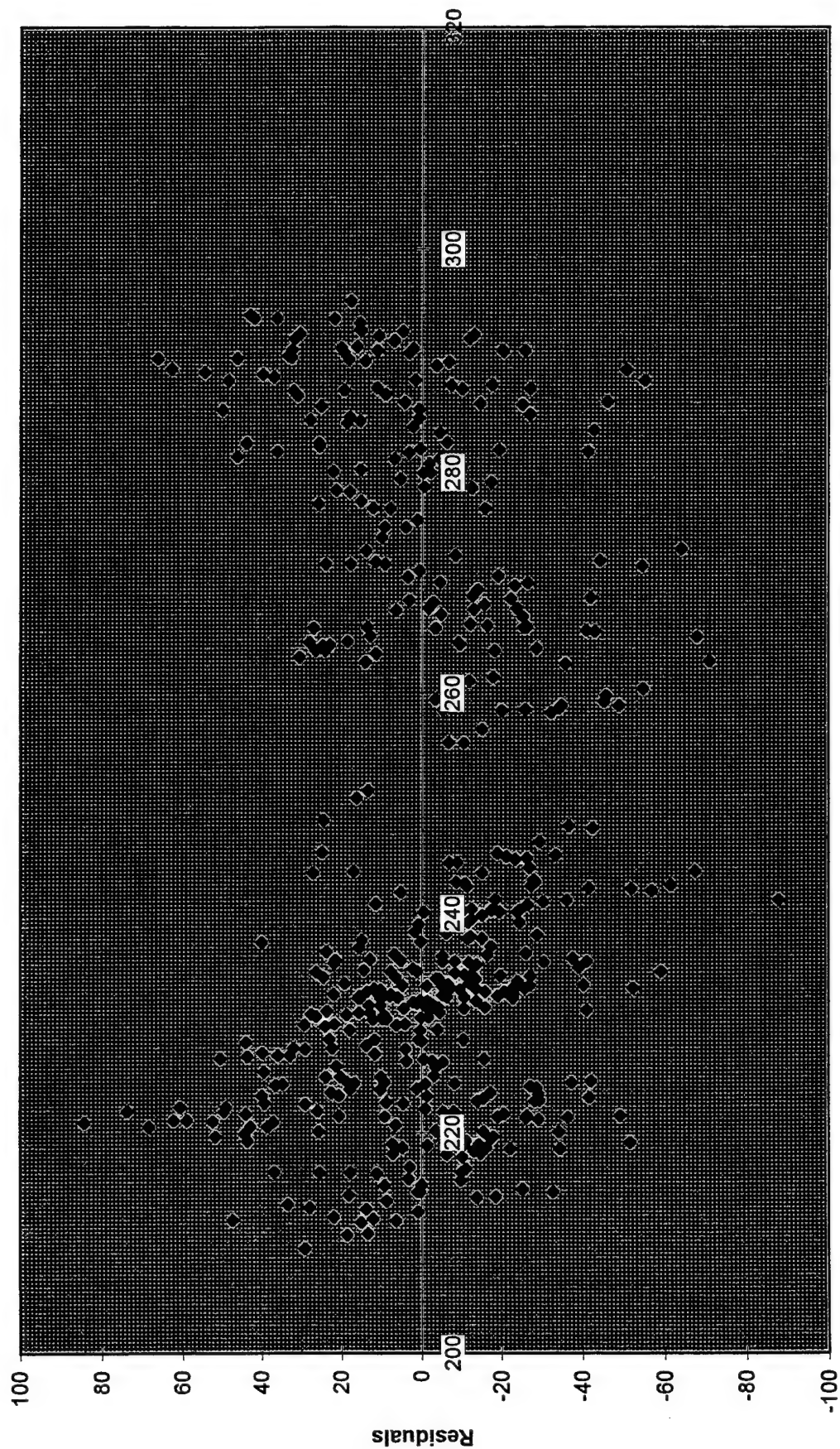


CRB C

Amex Oil Index C Residual Plot

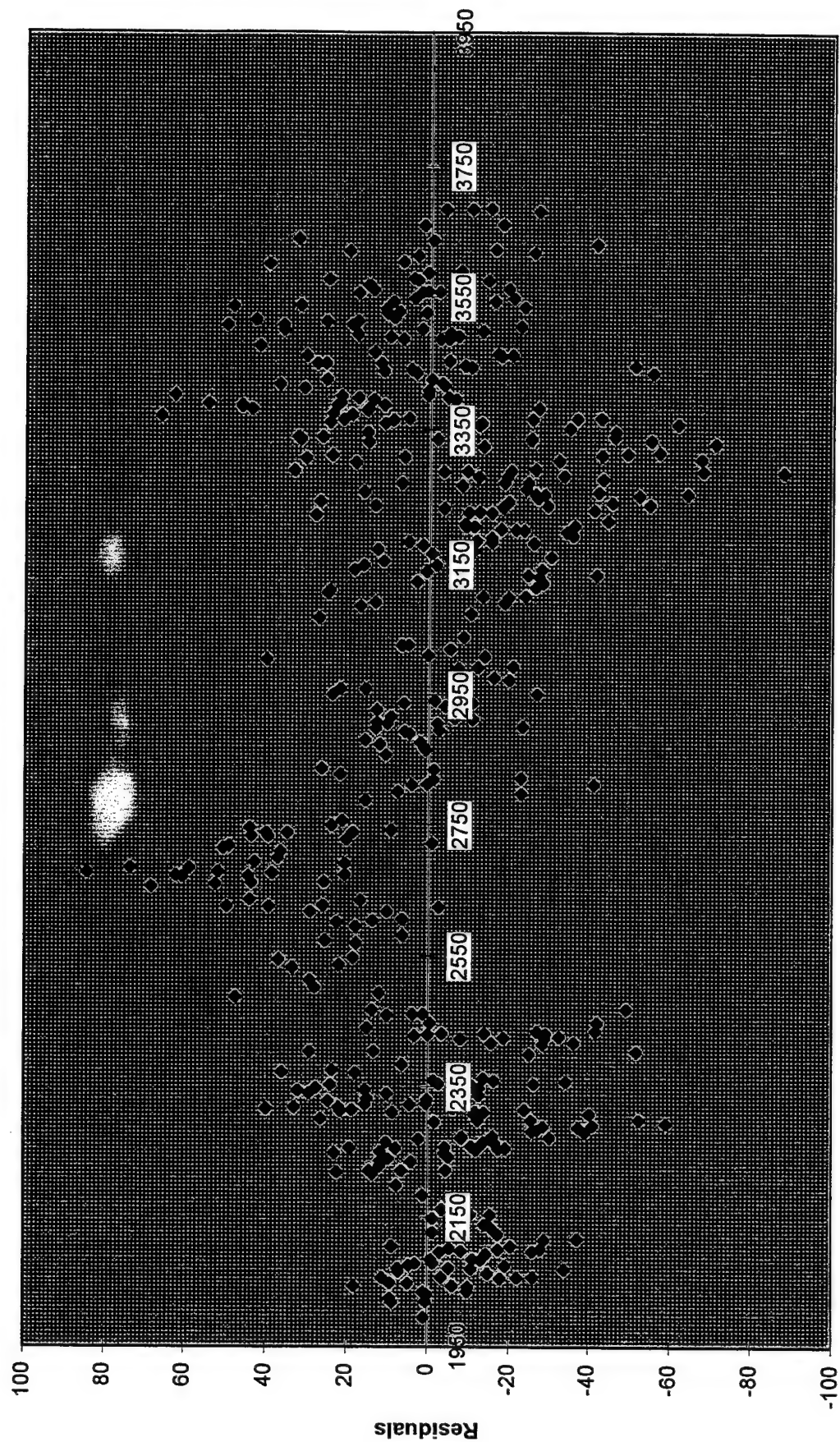


Dow Utilities Index C Residual Plot



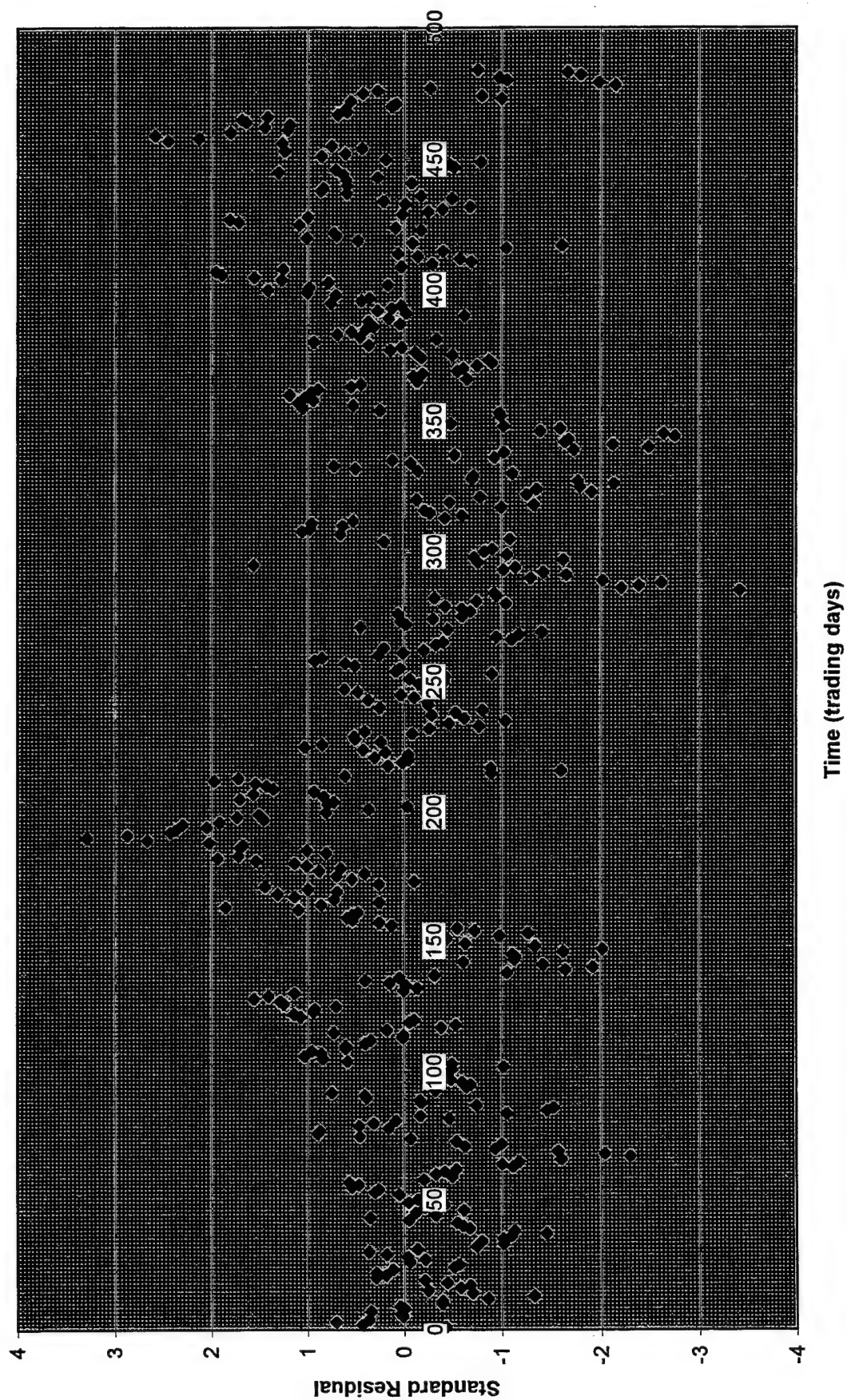
Dow Utilities Index C

Dow Transportation Index C Residual Plot

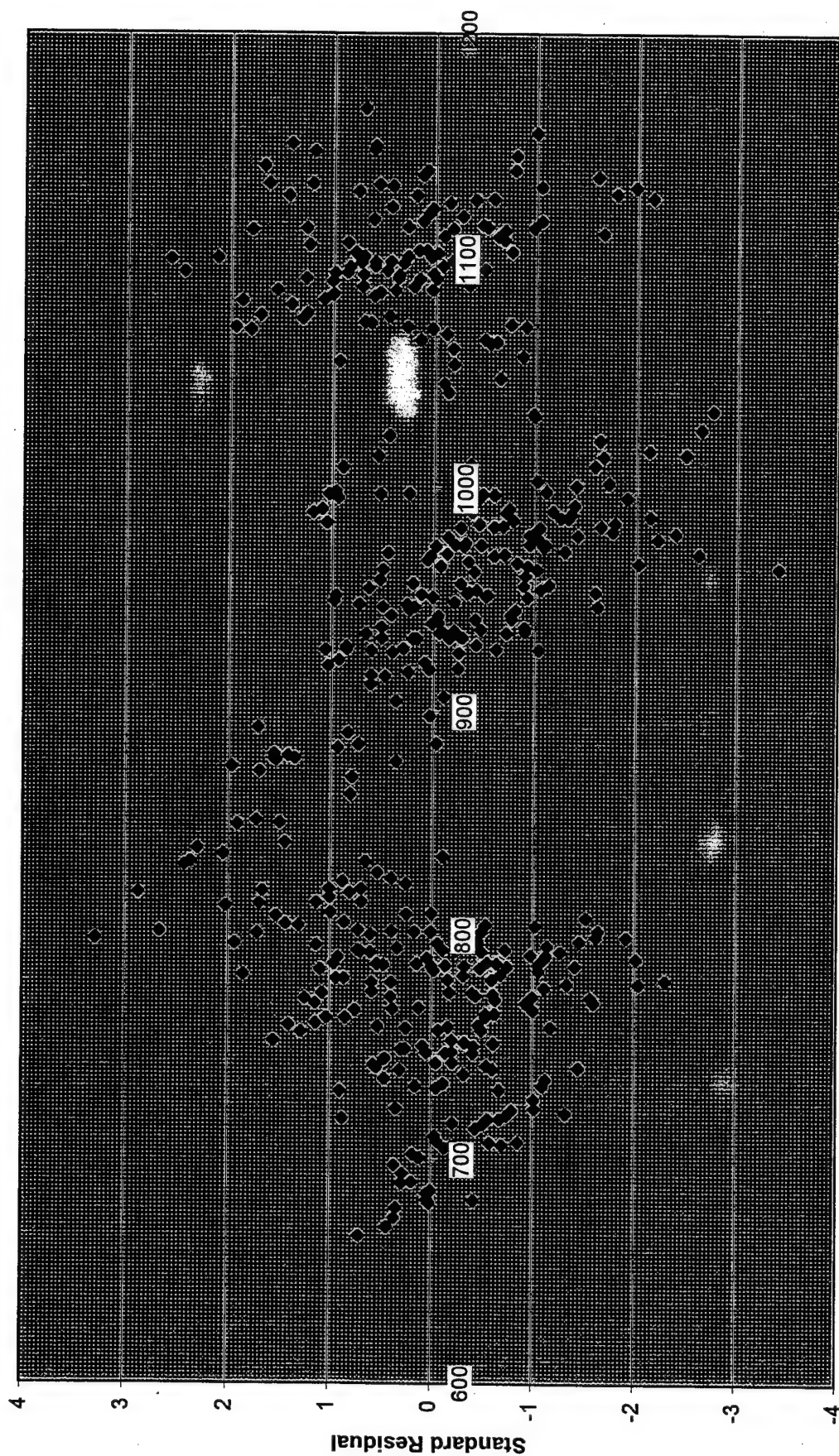


Dow Transportation Index C

Standard Residuals of Predicted S&P500 C vs Time



Predicted S&P500 C Standard Residual Plot



S&P500 Closing Value (\$)

APPENDIX B. PARTIAL LISTING OF RAW AND PREPROCESSED INPUT DATA

Raw Data

Date	S&P500 H	S&P500 L	S&P500 C	Dow Transportation Index C	Dow Utilities Index C	Amex Oil Index C	CRB C	Dow Industrial Index V	Gold and Silver Mining C
3/1/91	370.468	363.729	370.468	1150.739	213.339	252.5	217.94	272472	86.86
3/4/91	371.989	369.07	369.33	1142.449	212.02	250.34	219.69	214533	89.35
3/5/91	377.887	369.33	376.718	1166.26	213.85	253.58	220.82	283650	87.93
3/6/91	379.657	375.02	376.167	1151.62	211.639	253.97	220.809	312750	87.86
3/7/91	377.479	375.58	375.91	1141.57	211.139	254.62	220.91	212886	86.94
3/8/91	378.687	374.427	374.947	1125.179	210.449	254.4	221.839	243278	90.83
3/11/91	375.1	372.52	372.958	1119.36	211.639	251.78	218.99	165280	89.4
3/12/91	374.35	369.55	370.03	1112.479	212.649	251.99	219.949	182985	89.26
3/13/91	374.647	370.03	374.57	1111.419	213.22	253.9	220.77	190201	90.3
3/14/91	378.28	371.76	373.5	1107.37	211.77	251.82	219.88	250042	88.24
3/15/91	374.58	370.208	373.59	1098.729	212.46	251.64	217.929	383815	88.23
3/18/91	374.09	369.458	372.11	1099.79	211.71	248.33	217.52	170371	85.87
3/19/91	372.11	366.54	366.59	1084.8	211.08	249.34	218.72	252888	86.28
3/20/91	368.85	365.8	367.917	1087.449	212.02	250.38	218.639	257186	85.78
3/21/91	371.01	366.51	366.58	1084.8	212.02	250.13	219.86	237244	85.93
3/22/91	368.218	365.58	367.479	1075.459	213.85	250.85	219.779	174325	85.52
3/25/91	371.31	367.458	369.83	1075.28	216.36	250.59	217.809	167894	84.15
3/26/91	376.3	369.37	376.3	1102.79	218.119	251.13	217.279	256327	83.3
3/27/91	378.479	374.729	375.35	1117.949	216.55	248.11	217.339	252850	83.43
3/28/91	376.6	374.397	375.218	1109.489	217.179	246.68	218.47	178992	84.29
4/1/91	375.218	370.27	371.3	1101.55	213.91	244.92	221.619	162162	86.18
4/2/91	379.5	371.3	379.5	1115.479	216.74	248.15	222.199	210661	86.73
4/3/91	381.56	378.489	378.937	1116.179	216.3	246.25	221.49	206521	87.46
4/4/91	381.877	377.05	379.77	1121.649	216.11	244.49	220.72	221632	86.64
4/5/91	381.12	374.147	375.36	1114.949	215.1	242.04	220.449	201704	85.31
4/8/91	378.76	374.687	378.657	1123.239	216.169	244.31	221.509	139629	87.07
4/9/91	379.02	373.11	373.56	1111.6	214.47	243.34	220.85	173841	88.65
4/10/91	374.83	371.208	373.147	1113.889	214.979	244.16	221.139	170128	87.62
4/11/91	379.53	373.147	377.627	1121.12	217.3	249.23	220.66	239427	85.73
4/12/91	381.07	376.887	380.397	1128.879	218.179	253.86	220.71	229422	84.81
4/15/91	382.32	378.78	381.187	1123.239	218.809	256.49	220.58	218905	83.43
4/16/91	387.62	379.637	387.62	1143.51	220.889	258.68	220.57	257896	83.06
4/17/91	391.26	387.3	390.447	1166.26	219.759	259.86	220.16	327501	83.99
4/18/91	390.968	388.127	388.458	1178.07	216.3	258.39	218.039	252516	81.64
4/19/91	388.458	383.897	384.197	1172.07	212.649	256.63	217.72	237757	80.82
4/22/91	384.187	380.157	380.947	1151.62	210.949	254.98	217.789	211280	83.35
4/23/91	383.55	379.667	381.76	1157.09	209.509	257.71	217.949	185519	82.87
4/24/91	383.02	379.989	382.76	1171.02	210.389	258.43	218.139	202077	82.17
4/25/91	382.887	378.427	379.25	1170.129	209.38	255.26	217.72	174769	79.8
4/26/91	380.11	376.77	379.02	1166.08	209.82	255.12	217.74	173306	79.74
4/29/91	380.958	373.657	373.657	1148.8	208.309	253.47	216.149	163819	78.17

Preprocessed Data

[illegible]

APPENDIX C. SOURCE CODE FOR CLOSE NETWORK

Note - use this code with your calculator or spreadsheet to fire the
C:\24XLAN\JASON\SCHOOL\THESIS\SP500M~4\SP1NET~1\SP12NET\SP12NET
network

Note - the following are intermediate value cells and arrays:

Note - parens immediately below denote an array where the number of elements is in
parens

netsum
feature2(9)
feature3(9)
feature4(9)

Note - the following are names of inputs and outputs:

Note - inp(1) is S&P500_H

Note - inp(2) is S&P500_L

Note - inp(3) is S&P500_C

Note - inp(4) is Dow_Transportation_Index_C

Note - inp(5) is Dow_Uilities_Index_C

Note - inp(6) is Amex_Oil_Index_C

Note - inp(7) is CRB_C

Note - inp(8) is Gold_and_Silver_Mining_C

Note - inp(9) is GSM_C

Note - outp(1) is Future_(10_days)_S&P500_C

Note - inp(10) is MvAvg(30)_of_Dow_Industrial_Index_V

Note - inp(11) is Lag(10)_of_CRB_C

Note - inp(12) is Lag(10)_of_Amex_Oil_Index_C

Note - inp(13) is Lag(10)_of_Dow_Transportation_Index_C

Note - inp(14) is Lag(10)_of_GSM_C

if (inp(1)< 629) then inp(1) = 629

if (inp(1)> 1191) then inp(1) = 1191

inp(1) = 2 * (inp(1) - 629) / 562 -1

if (inp(2)< 616) then inp(2) = 616

if (inp(2)> 1182) then inp(2) = 1182

inp(2) = 2 * (inp(2) - 616) / 566 -1

if (inp(3)< 627) then inp(3) = 627

if (inp(3)> 1187) then inp(3) = 1187

inp(3) = 2 * (inp(3) - 627) / 560 -1

if (inp(4)< 1965.73) then inp(4) = 1965.73

if (inp(4)> 3686.02) then inp(4) = 3686.02

$\text{inp}(4) = 2 * (\text{inp}(4) - 1965.73) / 1720.29 - 1$

if $(\text{inp}(5) < 204.86)$ then $\text{inp}(5) = 204.86$

if $(\text{inp}(5) > 295.4)$ then $\text{inp}(5) = 295.4$

$\text{inp}(5) = 2 * (\text{inp}(5) - 204.86) / 90.53999 - 1$

if $(\text{inp}(6) < 330.48)$ then $\text{inp}(6) = 330.48$

if $(\text{inp}(6) > 503.75)$ then $\text{inp}(6) = 503.75$

$\text{inp}(6) = 2 * (\text{inp}(6) - 330.48) / 173.27 - 1$

if $(\text{inp}(7) < 202.02)$ then $\text{inp}(7) = 202.02$

if $(\text{inp}(7) > 253.96)$ then $\text{inp}(7) = 253.96$

$\text{inp}(7) = 2 * (\text{inp}(7) - 202.02) / 51.94 - 1$

if $(\text{inp}(8) < 60.32)$ then $\text{inp}(8) = 60.32$

if $(\text{inp}(8) > 129.33)$ then $\text{inp}(8) = 129.33$

$\text{inp}(8) = 2 * (\text{inp}(8) - 60.32) / 69.01 - 1$

if $(\text{inp}(9) < 78.24)$ then $\text{inp}(9) = 78.24$

if $(\text{inp}(9) > 198.23)$ then $\text{inp}(9) = 198.23$

$\text{inp}(9) = 2 * (\text{inp}(9) - 78.24) / 119.99 - 1$

if $(\text{inp}(10) < 277596.7)$ then $\text{inp}(10) = 277596.7$

if $(\text{inp}(10) > 713482.1)$ then $\text{inp}(10) = 713482.1$

$\text{inp}(10) = 2 * (\text{inp}(10) - 277596.7) / 435885.4 - 1$

if $(\text{inp}(11) < 206.14)$ then $\text{inp}(11) = 206.14$

if $(\text{inp}(11) > 253.96)$ then $\text{inp}(11) = 253.96$

$\text{inp}(11) = 2 * (\text{inp}(11) - 206.14) / 47.82001 - 1$

if $(\text{inp}(12) < 330.48)$ then $\text{inp}(12) = 330.48$

if $(\text{inp}(12) > 503.75)$ then $\text{inp}(12) = 503.75$

$\text{inp}(12) = 2 * (\text{inp}(12) - 330.48) / 173.27 - 1$

if $(\text{inp}(13) < 1965.73)$ then $\text{inp}(13) = 1965.73$

if $(\text{inp}(13) > 3686.02)$ then $\text{inp}(13) = 3686.02$

$\text{inp}(13) = 2 * (\text{inp}(13) - 1965.73) / 1720.29 - 1$

if $(\text{inp}(14) < 78.24)$ then $\text{inp}(14) = 78.24$

if $(\text{inp}(14) > 198.23)$ then $\text{inp}(14) = 198.23$

$\text{inp}(14) = 2 * (\text{inp}(14) - 78.24) / 119.99 - 1$

$\text{netsum} = -.6301715$

$\text{netsum} = \text{netsum} + \text{inp}(1) * .4015678$

$\text{netsum} = \text{netsum} + \text{inp}(2) * .1923304$

$\text{netsum} = \text{netsum} + \text{inp}(3) * .3034808$

```

netsum = netsum + inp(4) * .471424
netsum = netsum + inp(5) * -.522025
netsum = netsum + inp(6) * -.3444886
netsum = netsum + inp(7) * .2156871
netsum = netsum + inp(8) * -.0962563
netsum = netsum + inp(9) * 8.810111E-02
netsum = netsum + inp(10) * .8480121
netsum = netsum + inp(11) * 1.090916E-02
netsum = netsum + inp(12) * -.1067936
netsum = netsum + inp(13) * .192736
netsum = netsum + inp(14) * .2648861
feature2(1) = exp(-netsum * netsum)

```

```

netsum = 7.784155E-02
netsum = netsum + inp(1) * -.1590329
netsum = netsum + inp(2) * -6.405292E-02
netsum = netsum + inp(3) * .1000905
netsum = netsum + inp(4) * -.2733166
netsum = netsum + inp(5) * .2708781
netsum = netsum + inp(6) * -.2224024
netsum = netsum + inp(7) * -.6161205
netsum = netsum + inp(8) * .1161777
netsum = netsum + inp(9) * 9.140761E-02
netsum = netsum + inp(10) * .6487405
netsum = netsum + inp(11) * -6.036293E-02
netsum = netsum + inp(12) * 3.895962E-02
netsum = netsum + inp(13) * -.1787288
netsum = netsum + inp(14) * .333038
feature2(2) = exp(-netsum * netsum)

```

```

netsum = -.2859306
netsum = netsum + inp(1) * .2471741
netsum = netsum + inp(2) * .3450775
netsum = netsum + inp(3) * 8.962151E-02
netsum = netsum + inp(4) * 4.607785E-02
netsum = netsum + inp(5) * -6.669939E-02
netsum = netsum + inp(6) * -.1497742
netsum = netsum + inp(7) * .3083329
netsum = netsum + inp(8) * 8.747423E-02
netsum = netsum + inp(9) * -1.843768E-02
netsum = netsum + inp(10) * .2710629
netsum = netsum + inp(11) * .3533338
netsum = netsum + inp(12) * -.1935388
netsum = netsum + inp(13) * .1157563
netsum = netsum + inp(14) * -.3057612
feature2(3) = exp(-netsum * netsum)

```

```

netsum = .3307071
netsum = netsum + inp(1) * 7.340278E-02
netsum = netsum + inp(2) * 1.156442E-02
netsum = netsum + inp(3) * -.1988501
netsum = netsum + inp(4) * .1112011
netsum = netsum + inp(5) * 4.713702E-03
netsum = netsum + inp(6) * -.1553141
netsum = netsum + inp(7) * -.148536
netsum = netsum + inp(8) * .2287395
netsum = netsum + inp(9) * .1997013
netsum = netsum + inp(10) * -.4850825
netsum = netsum + inp(11) * -.7133386
netsum = netsum + inp(12) * .3967151
netsum = netsum + inp(13) * -.6160113
netsum = netsum + inp(14) * 3.598906E-03
feature2(4) = exp(-netsum * netsum)

```

```

netsum = 1.679392E-02
netsum = netsum + inp(1) * -5.007538E-03
netsum = netsum + inp(2) * -.237791
netsum = netsum + inp(3) * 8.680119E-02
netsum = netsum + inp(4) * .229306
netsum = netsum + inp(5) * -.1694554
netsum = netsum + inp(6) * .1752781
netsum = netsum + inp(7) * -.1325122
netsum = netsum + inp(8) * -3.663547E-02
netsum = netsum + inp(9) * -.1170656
netsum = netsum + inp(10) * -.2149827
netsum = netsum + inp(11) * .2217486
netsum = netsum + inp(12) * .1699766
netsum = netsum + inp(13) * -7.920565E-02
netsum = netsum + inp(14) * .1580922
feature2(5) = exp(-netsum * netsum)

```

```

netsum = 4.819342E-02
netsum = netsum + inp(1) * -5.559232E-02
netsum = netsum + inp(2) * -.2133055
netsum = netsum + inp(3) * .0223891
netsum = netsum + inp(4) * .1393356
netsum = netsum + inp(5) * .3481391
netsum = netsum + inp(6) * -.3444548
netsum = netsum + inp(7) * .1126457
netsum = netsum + inp(8) * .2515414
netsum = netsum + inp(9) * -.2600302
netsum = netsum + inp(10) * .1857713

```

```

netsum = netsum + inp(11) * .1466804
netsum = netsum + inp(12) * 8.555102E-02
netsum = netsum + inp(13) * -6.260171E-02
netsum = netsum + inp(14) * -.1186699
feature2(6) = exp(-netsum * netsum)

```

```

netsum = 1.077751E-03
netsum = netsum + inp(1) * -.31528
netsum = netsum + inp(2) * -.3444317
netsum = netsum + inp(3) * -.1208438
netsum = netsum + inp(4) * .450067
netsum = netsum + inp(5) * .0752024
netsum = netsum + inp(6) * 8.618691E-02
netsum = netsum + inp(7) * .1862327
netsum = netsum + inp(8) * 5.465092E-02
netsum = netsum + inp(9) * -.1758856
netsum = netsum + inp(10) * -.1402889
netsum = netsum + inp(11) * -5.002017E-02
netsum = netsum + inp(12) * -8.830319E-02
netsum = netsum + inp(13) * 4.461126E-02
netsum = netsum + inp(14) * -.1783304
feature2(7) = exp(-netsum * netsum)

```

```

netsum = .2010598
netsum = netsum + inp(1) * -.2251198
netsum = netsum + inp(2) * -5.359336E-02
netsum = netsum + inp(3) * -.2414816
netsum = netsum + inp(4) * .2557169
netsum = netsum + inp(5) * .3187351
netsum = netsum + inp(6) * -.2540528
netsum = netsum + inp(7) * .1812858
netsum = netsum + inp(8) * -.2384627
netsum = netsum + inp(9) * -.0457608
netsum = netsum + inp(10) * -.2491588
netsum = netsum + inp(11) * -5.906731E-02
netsum = netsum + inp(12) * -.2001718
netsum = netsum + inp(13) * .120379
netsum = netsum + inp(14) * .1183008
feature2(8) = exp(-netsum * netsum)

```

```

netsum = .112657
netsum = netsum + inp(1) * -.1262439
netsum = netsum + inp(2) * -5.584108E-02
netsum = netsum + inp(3) * 5.810664E-02
netsum = netsum + inp(4) * 4.742816E-02
netsum = netsum + inp(5) * .1374272

```

```

netsum = netsum + inp(6) * .1569632
netsum = netsum + inp(7) * .1107642
netsum = netsum + inp(8) * .1762077
netsum = netsum + inp(9) * .228175
netsum = netsum + inp(10) * 7.341717E-02
netsum = netsum + inp(11) * -.3403822
netsum = netsum + inp(12) * .2044954
netsum = netsum + inp(13) * 9.120224E-02
netsum = netsum + inp(14) * -.303437
feature2(9) = exp(-netsum * netsum)

```

```

netsum = -.1239542
netsum = netsum + inp(1) * 6.883536E-02
netsum = netsum + inp(2) * .1504488
netsum = netsum + inp(3) * -.1053862
netsum = netsum + inp(4) * -.1535371
netsum = netsum + inp(5) * -.2344521
netsum = netsum + inp(6) * -.204491
netsum = netsum + inp(7) * 8.043891E-02
netsum = netsum + inp(8) * 7.446315E-02
netsum = netsum + inp(9) * -3.834013E-02
netsum = netsum + inp(10) * .2165757
netsum = netsum + inp(11) * -.1488486
netsum = netsum + inp(12) * 5.801564E-02
netsum = netsum + inp(13) * .2320869
netsum = netsum + inp(14) * .2076355
feature3(1) = tanh(netsum)

```

```

netsum = .3126868
netsum = netsum + inp(1) * .1616008
netsum = netsum + inp(2) * 3.386919E-02
netsum = netsum + inp(3) * -.1983408
netsum = netsum + inp(4) * -.3485649
netsum = netsum + inp(5) * -.1106689
netsum = netsum + inp(6) * -.1445011
netsum = netsum + inp(7) * .3310095
netsum = netsum + inp(8) * -.2066913
netsum = netsum + inp(9) * -.2082781
netsum = netsum + inp(10) * 5.758699E-02
netsum = netsum + inp(11) * -6.427113E-02
netsum = netsum + inp(12) * 2.970282E-02
netsum = netsum + inp(13) * -.2802488
netsum = netsum + inp(14) * .252078
feature3(2) = tanh(netsum)

```

```

netsum = 4.668346E-02

```

```

netsum = netsum + inp(1) * -.1714819
netsum = netsum + inp(2) * -6.209589E-02
netsum = netsum + inp(3) * -.1907481
netsum = netsum + inp(4) * -.1376314
netsum = netsum + inp(5) * .2418897
netsum = netsum + inp(6) * -.2630582
netsum = netsum + inp(7) * -.2526979
netsum = netsum + inp(8) * -.1087746
netsum = netsum + inp(9) * 7.395954E-02
netsum = netsum + inp(10) * .1435177
netsum = netsum + inp(11) * .1491603
netsum = netsum + inp(12) * .1827831
netsum = netsum + inp(13) * .3284236
netsum = netsum + inp(14) * .1364697
feature3(3) = tanh(netsum)

```

```

netsum = 3.868202E-02
netsum = netsum + inp(1) * .402357
netsum = netsum + inp(2) * .2806031
netsum = netsum + inp(3) * .2690603
netsum = netsum + inp(4) * 6.963006E-02
netsum = netsum + inp(5) * .1762392
netsum = netsum + inp(6) * .1347711
netsum = netsum + inp(7) * -.3312288
netsum = netsum + inp(8) * .4110459
netsum = netsum + inp(9) * .182495
netsum = netsum + inp(10) * 7.211982E-02
netsum = netsum + inp(11) * 7.574815E-02
netsum = netsum + inp(12) * .3440028
netsum = netsum + inp(13) * .4007721
netsum = netsum + inp(14) * -.2247503
feature3(4) = tanh(netsum)

```

```

netsum = 1.236538E-02
netsum = netsum + inp(1) * 1.840403E-02
netsum = netsum + inp(2) * -.2597425
netsum = netsum + inp(3) * .2060692
netsum = netsum + inp(4) * .1866937
netsum = netsum + inp(5) * 8.070311E-03
netsum = netsum + inp(6) * 6.152835E-02
netsum = netsum + inp(7) * -.183747
netsum = netsum + inp(8) * -.301769
netsum = netsum + inp(9) * -3.505382E-02
netsum = netsum + inp(10) * -4.599997E-02
netsum = netsum + inp(11) * 2.352656E-02
netsum = netsum + inp(12) * -.1942575

```



```
netsum = netsum + inp(13) * -.1482109
netsum = netsum + inp(14) * -8.374985E-02
feature3(5) = tanh(netsum)
```

```
netsum = -.2077845
netsum = netsum + inp(1) * -.3183326
netsum = netsum + inp(2) * -.2343775
netsum = netsum + inp(3) * 2.819597E-02
netsum = netsum + inp(4) * .1284694
netsum = netsum + inp(5) * -.2394804
netsum = netsum + inp(6) * -4.67339E-03
netsum = netsum + inp(7) * .6174434
netsum = netsum + inp(8) * .4019511
netsum = netsum + inp(9) * 5.051733E-02
netsum = netsum + inp(10) * -.3909052
netsum = netsum + inp(11) * 7.491957E-02
netsum = netsum + inp(12) * 5.784418E-02
netsum = netsum + inp(13) * -.0298114
netsum = netsum + inp(14) * -.1702783
feature3(6) = tanh(netsum)
```

```
netsum = .1813595
netsum = netsum + inp(1) * 1.982817E-02
netsum = netsum + inp(2) * .1559182
netsum = netsum + inp(3) * .0207838
netsum = netsum + inp(4) * -.2390079
netsum = netsum + inp(5) * -8.514214E-02
netsum = netsum + inp(6) * -.2670723
netsum = netsum + inp(7) * 1.109988E-02
netsum = netsum + inp(8) * -7.678013E-02
netsum = netsum + inp(9) * 9.093736E-02
netsum = netsum + inp(10) * .1590883
netsum = netsum + inp(11) * .1004014
netsum = netsum + inp(12) * .2035837
netsum = netsum + inp(13) * -6.611361E-02
netsum = netsum + inp(14) * -.1934318
feature3(7) = tanh(netsum)
```

```
netsum = .2397276
netsum = netsum + inp(1) * -.1992782
netsum = netsum + inp(2) * -.128719
netsum = netsum + inp(3) * .1892586
netsum = netsum + inp(4) * 9.737152E-02
netsum = netsum + inp(5) * -.1388893
netsum = netsum + inp(6) * 8.414371E-02
netsum = netsum + inp(7) * .4454928
```

```

netsum = netsum + inp(8) * -.2841288
netsum = netsum + inp(9) * 5.766987E-02
netsum = netsum + inp(10) * 5.085299E-02
netsum = netsum + inp(11) * -.2496866
netsum = netsum + inp(12) * .3022949
netsum = netsum + inp(13) * -.3111849
netsum = netsum + inp(14) * .1354086
feature3(8) = tanh(netsum)

```

```

netsum = -5.716069E-02
netsum = netsum + inp(1) * .1918543
netsum = netsum + inp(2) * .2528775
netsum = netsum + inp(3) * -.1584775
netsum = netsum + inp(4) * -.2948896
netsum = netsum + inp(5) * .1197234
netsum = netsum + inp(6) * .3107863
netsum = netsum + inp(7) * -.217381
netsum = netsum + inp(8) * .1799645
netsum = netsum + inp(9) * .2089709
netsum = netsum + inp(10) * .2344017
netsum = netsum + inp(11) * .142157
netsum = netsum + inp(12) * .0758443
netsum = netsum + inp(13) * -.2458397
netsum = netsum + inp(14) * .1971597
feature3(9) = tanh(netsum)

```

```

netsum = .2587757
netsum = netsum + inp(1) * 6.334911E-02
netsum = netsum + inp(2) * -.1395912
netsum = netsum + inp(3) * .2138192
netsum = netsum + inp(4) * -.1274552
netsum = netsum + inp(5) * -.1393314
netsum = netsum + inp(6) * -.2645777
netsum = netsum + inp(7) * .2876672
netsum = netsum + inp(8) * -.1303833
netsum = netsum + inp(9) * .2642182
netsum = netsum + inp(10) * 3.467115E-02
netsum = netsum + inp(11) * .1072136
netsum = netsum + inp(12) * -.1934204
netsum = netsum + inp(13) * -.1403725
netsum = netsum + inp(14) * -.1843849
feature4(1) = 1 - exp(-netsum * netsum)

```

```

netsum = .2993304
netsum = netsum + inp(1) * -8.428704E-03
netsum = netsum + inp(2) * -.1071872

```

```

netsum = netsum + inp(3) * -8.796584E-02
netsum = netsum + inp(4) * .1266784
netsum = netsum + inp(5) * .2506436
netsum = netsum + inp(6) * 4.853629E-02
netsum = netsum + inp(7) * -.4751402
netsum = netsum + inp(8) * -.136122
netsum = netsum + inp(9) * -.3058958
netsum = netsum + inp(10) * -5.209611E-02
netsum = netsum + inp(11) * .3099503
netsum = netsum + inp(12) * -.1840532
netsum = netsum + inp(13) * .3218807
netsum = netsum + inp(14) * .1453303
feature4(2) = 1 - exp(-netsum * netsum)

```

```

netsum = .1933935
netsum = netsum + inp(1) * .3332409
netsum = netsum + inp(2) * .2873065
netsum = netsum + inp(3) * 1.045562E-03
netsum = netsum + inp(4) * .2716179
netsum = netsum + inp(5) * 3.613625E-02
netsum = netsum + inp(6) * -.3512647
netsum = netsum + inp(7) * -.301256
netsum = netsum + inp(8) * -6.802244E-02
netsum = netsum + inp(9) * 5.407644E-02
netsum = netsum + inp(10) * 7.879863E-03
netsum = netsum + inp(11) * 5.932507E-02
netsum = netsum + inp(12) * 3.043652E-02
netsum = netsum + inp(13) * -.1559501
netsum = netsum + inp(14) * -.1701026
feature4(3) = 1 - exp(-netsum * netsum)

```

```

netsum = .1759639
netsum = netsum + inp(1) * -.6680542
netsum = netsum + inp(2) * -.2973823
netsum = netsum + inp(3) * -.3279465
netsum = netsum + inp(4) * -.1164177
netsum = netsum + inp(5) * -.6758547
netsum = netsum + inp(6) * -5.386759E-02
netsum = netsum + inp(7) * .4030443
netsum = netsum + inp(8) * -.337681
netsum = netsum + inp(9) * -.2303785
netsum = netsum + inp(10) * -.4204697
netsum = netsum + inp(11) * .2685212
netsum = netsum + inp(12) * .3571743
netsum = netsum + inp(13) * -.3708806
netsum = netsum + inp(14) * -.4410872

```

feature4(4) = 1 - exp(-netsum * netsum)

netsum = -7.085525E-02

netsum = netsum + inp(1) * .2006301

netsum = netsum + inp(2) * 9.633981E-02

netsum = netsum + inp(3) * -.0126803

netsum = netsum + inp(4) * .1226519

netsum = netsum + inp(5) * -8.257607E-02

netsum = netsum + inp(6) * -.2301209

netsum = netsum + inp(7) * .1052367

netsum = netsum + inp(8) * .230889

netsum = netsum + inp(9) * -.1533706

netsum = netsum + inp(10) * -.1561237

netsum = netsum + inp(11) * 8.230709E-02

netsum = netsum + inp(12) * -.3135089

netsum = netsum + inp(13) * -.2566093

netsum = netsum + inp(14) * .3265928

feature4(5) = 1 - exp(-netsum * netsum)

netsum = .2666591

netsum = netsum + inp(1) * -.163848

netsum = netsum + inp(2) * -.633084

netsum = netsum + inp(3) * -.1689773

netsum = netsum + inp(4) * -.5006466

netsum = netsum + inp(5) * -.419012

netsum = netsum + inp(6) * 4.745518E-02

netsum = netsum + inp(7) * -.2422797

netsum = netsum + inp(8) * .4093167

netsum = netsum + inp(9) * -5.734261E-02

netsum = netsum + inp(10) * -.3980823

netsum = netsum + inp(11) * 1.593639E-02

netsum = netsum + inp(12) * -.2957138

netsum = netsum + inp(13) * -7.563066E-03

netsum = netsum + inp(14) * -8.185922E-03

feature4(6) = 1 - exp(-netsum * netsum)

netsum = -.1909794

netsum = netsum + inp(1) * -.1831633

netsum = netsum + inp(2) * -4.450019E-02

netsum = netsum + inp(3) * -.1683403

netsum = netsum + inp(4) * -.1905011

netsum = netsum + inp(5) * .1881501

netsum = netsum + inp(6) * .4262969

netsum = netsum + inp(7) * -.2553455

netsum = netsum + inp(8) * .1917894

netsum = netsum + inp(9) * .3827554

```

netsum = netsum + inp(10) * -3.722818E-02
netsum = netsum + inp(11) * .4479087
netsum = netsum + inp(12) * 2.176078E-02
netsum = netsum + inp(13) * -.295205
netsum = netsum + inp(14) * .1430879
feature4(7) = 1 - exp(-netsum * netsum)

```

```

netsum = -.2656472
netsum = netsum + inp(1) * -.2388305
netsum = netsum + inp(2) * -7.188458E-02
netsum = netsum + inp(3) * -.297852
netsum = netsum + inp(4) * .1506731
netsum = netsum + inp(5) * .2484149
netsum = netsum + inp(6) * 5.983697E-02
netsum = netsum + inp(7) * -.1939304
netsum = netsum + inp(8) * 6.954922E-02
netsum = netsum + inp(9) * -.2615168
netsum = netsum + inp(10) * -.2277467
netsum = netsum + inp(11) * -.1542342
netsum = netsum + inp(12) * -6.928023E-02
netsum = netsum + inp(13) * -.2817874
netsum = netsum + inp(14) * .207139
feature4(8) = 1 - exp(-netsum * netsum)

```

```

netsum = .1556563
netsum = netsum + inp(1) * -6.293206E-03
netsum = netsum + inp(2) * -1.832035E-03
netsum = netsum + inp(3) * .1134438
netsum = netsum + inp(4) * 1.824013E-02
netsum = netsum + inp(5) * 5.927859E-02
netsum = netsum + inp(6) * .1220353
netsum = netsum + inp(7) * 7.248743E-02
netsum = netsum + inp(8) * .1827504
netsum = netsum + inp(9) * -2.891132E-02
netsum = netsum + inp(10) * -.1420365
netsum = netsum + inp(11) * .2402575
netsum = netsum + inp(12) * -.2662137
netsum = netsum + inp(13) * -.2764541
netsum = netsum + inp(14) * .1470288
feature4(9) = 1 - exp(-netsum * netsum)

```

```

netsum = -.2403532
netsum = netsum + feature2(1) * .8336458
netsum = netsum + feature2(2) * .7772395
netsum = netsum + feature2(3) * -.3676423
netsum = netsum + feature2(4) * -.8560631

```

```

netsum = netsum + feature2(5) * -3.738723E-03
netsum = netsum + feature2(6) * -.2726331
netsum = netsum + feature2(7) * .4049881
netsum = netsum + feature2(8) * -9.708894E-02
netsum = netsum + feature2(9) * -.3385944
netsum = netsum + 2.389552E-02
netsum = netsum + feature3(1) * -.2318477
netsum = netsum + feature3(2) * -9.119899E-02
netsum = netsum + feature3(3) * -.3589992
netsum = netsum + feature3(4) * .4774115
netsum = netsum + feature3(5) * .1890002
netsum = netsum + feature3(6) * -.6112313
netsum = netsum + feature3(7) * -.1102381
netsum = netsum + feature3(8) * .4290453
netsum = netsum + feature3(9) * -6.637502E-02
netsum = netsum + -.0645455
netsum = netsum + feature4(1) * -.2677609
netsum = netsum + feature4(2) * -.4827854
netsum = netsum + feature4(3) * .2690243
netsum = netsum + feature4(4) * .9494663
netsum = netsum + feature4(5) * .2479254
netsum = netsum + feature4(6) * -.5345625
netsum = netsum + feature4(7) * .5549127
netsum = netsum + feature4(8) * 6.048515E-03
netsum = netsum + feature4(9) * .1529618
outp(1) = 1 / (1 + exp(-netsum))

```

```

outp(1) = 538 * (outp(1) - .1) / .8 + 649

```


LIST OF REFERENCES

- Beale, R.; Jackson T., *Neural Computing: An Introduction*, Adam Hilger, Bristol, England, 1990.
- Dhar, Vasant, Stein, Roger, ed., *Intelligent Decision Support Methods: The Science of Knowledge Work*, Prentice Hall Business Publishing, Upper Saddle River, New Jersey, 1996.
- Gately, Edward J., *Neural Networks for Financial Forecasting*, John Wiley & Sons, New York, New York, 1996.
- Haykin, Simon, *Neural Networks: A Comprehensive Foundation*, Macmillian College Publishing Company, New York, New York, 1994.
- Kartalopoulos, Stamatios V., *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications*, IEEE Press, New York, New York, 1996.
- Lederman, Jess, Klein, Robert A., ed., *Virtual Trading: How Any Trader With a PC Can Use the Power of Neural Networks and Expert Systems to Boost Trading Profits*, Irwin Professional Publishing, New York, New York, 1995.
- Lowe, David, "Novel Exploitation of Neural Network Methods in Financial Markets," *Proceedings of the 3rd IEE International Conference on Artificial Neural Networks*, IEE Publications, Aston, United Kingdom, 1994.
- Smolensky, Paul, Mozer, Michael C., Rumelhart David E., ed., *Mathematical Perspectives on Neural Networks*, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1996.
- Ward, Steven, Sherald, Marge, "The Neural Network Financial Wizards," *Technical Analysis of Stocks & Commodities*, Reprinted, Technical Analysis Inc., Seattle, Washington, 1995.

BIBLIOGRAPHY

Braspenning P.J., Thuijsman, F., Weijters, A.J.M.M., ed., *Artificial Neural Networks: An Introduction to ANN Theory and Practice*, Springer, Hiedelberg, Germany, 1991.

Edwards, Robert D., Magee, John, *Technical Analysis of Stock Trends*, John Magee, Springfield Massachusetts, 1966.

Know, Harvey A., *Stock Market Behavior—Technical Approach to Understanding Wall Street*, Random House, New York, New York, 1969.

Sarle, W.S., ed., (1998), Neural Network FAQ, periodic posting to the Usenet newsgroup comp.ai.neural-nets, [URL:ftp://ftp.sas.com/pub/neural/FAQ.html](ftp://ftp.sas.com/pub/neural/FAQ.html)

Ward, Steven, Sherald, Marge, ed., *NeuroShell 2 Users Manual*, Ward Systems Group, Inc., Frederick, Maryland, 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
8725 John J. Kingman Road, Suite 0944
Fort Belvoir, VA 22060-6218
2. Dudley Knox Library2
Naval Postgraduate School
411 Dyer Road
Monterey, CA 93943-5101
3. Dr. Katsuaki Terasawa1
787 Shady Oaks Circle
Oxford, MS 38655
4. Dr. William R. Gates (Code SM/Gt)1
Naval Postgraduate School
555 Dyer Road
Monterey, CA 93943-5103
5. Editor, Stocks and Commodities1
4757 California Ave. SW
Seattle, WA 98116-4499
6. Mr. Steve Ward and Ms. Marge Sherald1
Ward Systems Group, Inc.
5 Hillcrest Drive
Frederick, MD 21702
7. Mr. Murray A. Ruggiero, Jr.1
Ruggiero & Associates
18 Oregon Ave.
East Haven, CT 06512
8. Mr. Milton Kutsurelis1
203 Crestview Drive
Nampa, ID 83686
9. LT Jason Kutsurelis2
348 Ardennes Circle
Seaside, CA 93955